F-SVR: A NEW LEARNING ALGORITHM FOR SUPPORT VECTOR REGRESSION

Mireille Tohmé[(1)]*, Régis Lengellé*[(2)]

[(1)] FORENAP Frp, 27, Rue du 4eme RSM , 68250 Rouffach, France

[(2)]ICD-LM2S (FRE CNRS 2848), Troyes University of Technology, BP 2060, 10010 Troyes cedex, France

ABSTRACT

In this paper, we present a new method for optimizing Support Vector Machines for regression problems. This algorithm searches for efficient feasible directions. Within these selected directions, we choose the best one, i.e. the one, coupled with an optimal step analytical evaluation, that ensures a maximum increase of the objective function. The resulting solution, the gradient and the objective function are recursively determined and the Gram matrix has not to be stored. Our algorithm is based on SVM-Torch proposed by Collobert for regression, which is similar to SVM-Light suggested by Joachims for classifications problems, but adapted to regression problems. We are also inspired by LASVM proposed by Bordes for classification problems. F-SVR algorithm uses a new efficient working set selection heuristic, ingeniously exploits quadratic function properties, so it is fast as well as easy to implement and is able to perform on large data sets.

Index Terms— Training, algorithm, Support Vector Machines

1. INTRODUCTION

Over the past years, Support Vector Machine (SVM) have become an important tool for solving pattern-recognition and regression problems. Recently, Smola and Scholkopf [1] proposed an iterative algorithm, called Sequential Minimal Optimization (SMO), for solving the regression problem using SVM. Their decomposition algorithm used an analytical solution for the subproblems, and they proposed to select two pairs of variables as a working set. This algorithm is an extension of the SMO proposed by Platt [2] for SVM classification. Bordes [3] suggested LASVM which is a reorganization of the SMO sequential direction search method and, as such, converges to the solution of the SVM quadratic programming problem. Collobert et al [4] used a decomposition algorithm similar to the one proposed by Joachims [5]. They suggested to select two variables instead of two pairs of variables as did Scholkopf and Smola. In fact, working with pairs of variables would force the algorithm to do many computations with null variables until the end of the optimization process. In this paper, we take advantage of the decomposition method to present a new learning algorithm for solving regression problems. F-SVR relies on the following four points:

1. The particular analytical expression of the equality constraint of the optimization problem
2. An efficient and new heuristic for selecting a possible working set
3. The analytical determination of the optimal step-size of the gradient algorithm which, coupled with the previous heuristic, is new within the context of SVM training
4. The analytical recursive computation of the solution, the gradient and the objective function

The two last properties result from the quadratic nature of the objective function, ensure the efficacy of F-SVR algorithm and make the difference with traditional algorithms. We begin this paper with a brief review of the basic technique used for implementing Support Vector Machines for regression problems. Then we give an overview of F-SVR by introducing in details its different steps. Section 4 presents the experimental results and we finally conclude with some perspectives.

2. SUPPORT VECTOR REGRESSION

In this section, we concisely review the fundamentals of Support Vector Regression in the non linear case. As usual, suppose we are given training data $(x_1, y_1), \cdots, (x_N, y_N) \subset \mathcal{X} \times \mathbb{R}$ where $\mathcal{X}$ denotes the space of the input patterns, for example $\mathbb{R}^d$. In ε-SV regression [6], Vapnik introduced the ε-insensitive loss function that enables sparsity within the Support Vectors (SV). Our goal is to find a function $f(x)$ that has at most ε deviation from the actually obtained targets y_i for all the training data, and at the same time is as smooth as possible. In other words, we do not care about errors as long as they are less than ε, but will pay for any deviation larger than ε. So we need to construct a regression function

$$\mathrm{f}(x) = \langle w, \Phi(x) \rangle + b \qquad w \in \Phi(\mathcal{X}), b \in \mathbb{R} \qquad (1)$$

on a feature space $F = \Phi(\mathcal{X})$. Here, w is a vector in F, and $\Phi(x)$ maps the input x to a vector in F. The w and b in (1) are obtained by solving the following quadratic optimization

problem:

$$\begin{array}{l} \underset{w,b,\xi,\xi^*}{\text{minimize}} \ \ \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ \text{subject to} \left\{ \begin{array}{l} y_i - \left(\left\langle w, \Phi(\mathrm{x})\right\rangle + b\right) \leq \varepsilon + \xi_i \\ \left(\left\langle w, \Phi(\mathrm{x})\right\rangle + b\right) - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0, \qquad i = 1, ..., N \end{array} \right. \end{array} \tag{2}$$

The constant $C > 0$ determines the trade-off between the smoothness of f and the cost associated to deviations larger than ε and ξ, ξ^* are the usual slack variables. The optimization problem (2) can be solved in dual form by introducing the positive Lagrange multipliers $\alpha, \alpha^*, \eta, \eta^*$. We can write the Lagrangian equation as follows:

$$\begin{array}{rl} L = & \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) - \sum_{i=1}^{N}(\eta_i\xi_i + \eta_i^*\xi_i^*) - \\ & \sum_{i=1}^{N}\alpha_i\left(\varepsilon + \xi_i + y_i - \left(\left\langle w, \Phi(\mathrm{x_i})\right\rangle + b\right)\right) - \\ & \sum_{i=1}^{N}\alpha_i^*\left(\varepsilon + \xi_i^* - y_i + \left(\left\langle w, \Phi(\mathrm{x})\right\rangle + b\right)\right) \end{array} \tag{3}$$

When minimizing L with respect to the primal variables (w, b, ξ_i, ξ_i^*), the derivative has to vanish. It can be shown that the dual variables η_i, η_i^* are eliminated, so we obtain the dual optimization problem:

$$\begin{array}{l} \underset{\alpha,\alpha^*}{\max} \quad W(\alpha^{(*)}) = -\frac{1}{2}\sum_{i,j=1}^{N}(\alpha_i - \alpha_i^*)\left(\alpha_j - \alpha_j^*\right)k(x_i, x_j) \\ \qquad\qquad -\varepsilon\sum_{i=1}^{N}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{N} y_i(\alpha_i - \alpha_i^*) \\ \text{subject} \quad \text{to} \left\{ \begin{array}{l} \sum_{i=1}^{N}(\alpha_i - \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \quad \in [0, C] \end{array} \right. \end{array} \tag{4}$$

where $\alpha^{(*)} = (\alpha_1, \ldots, \alpha_N, \alpha_1^*, \ldots, \alpha_N^*)^T$ and $k(x_i, x_j) = \left\langle \Phi(x_i), \Phi(x_j)\right\rangle$. The estimate of the regression function at any given point x is then:

$$f(x) = \sum_{i=1}^{N}(\alpha_i - \alpha_i^*)k(x_i, x) + b \tag{5}$$

where b is computed by exploiting the Karush-Kuhn-Tucker (KKT) conditions.

3. ALGORITHM DESCRIPTION

In this section, we present an overview of our proposed F-SVR learning algorithm for Support Vector Regression. The goal of the regression problem is to find the Lagrange multipliers $\alpha^{(*)}$, solution of the optimization problem (4). To solve this problem, we consider the particular analytical expression of the linear constraint in (4). At least two components of α and/or α^* must be modified. We shall consider here the simplest case where only two components will be updated. Examining this linear constraint leads to four possible cases, where $\lambda > 0$ is the only possibility to consider:

- case 1: $\alpha_i \leftarrow \alpha_i + \lambda$; $\alpha_j \leftarrow \alpha_j - \lambda$
- case 2: $\alpha_i^* \leftarrow \alpha_i^* + \lambda$; $\alpha_j^* \leftarrow \alpha_j^* - \lambda$
- case 3: $\alpha_i \leftarrow \alpha_i + \lambda$; $\alpha_j^* \leftarrow \alpha_j^* + \lambda$
- case 4: $\alpha_i \leftarrow \alpha_i - \lambda$; $\alpha_j^* \leftarrow \alpha_j^* - \lambda$

3.1. Searching for the *best* couple for each case (possible working set)

For every case, the modification of the selected components of α and/or α^* is defined by a vector $u^{(*)} = (u_1, \ldots, u_N, u_1^*, \ldots, u_N^*)^T$ that has only two non-null components equal to ± 1. In [5], it has been proposed to select the direction $u^{(*)}$ in such a way that ${u^{(*)}}^T g^{(*)}$ is maximum, where $g^{(*)} = (g_1, \ldots, g_N, g_1^*, \ldots, g_N^*)^T$ is the gradient of the objective function $W(\alpha^{(*)})$. This is exactly what we use here:

- case 1: $(i_1, j_1) = \underset{i,j}{\arg\max}\,(g_i - g_j) \ \Leftrightarrow$
  $i_1 = \underset{i}{\arg\max}(g_i); j_1 = \underset{j}{\arg\min}(g_j)$
  $(u_{i1} = +1, u_{j1} = -1)$
- case 2: $(i_2, j_2) = \underset{i,j}{\arg\max}\left(g_i^* - g_j^*\right) \ \Leftrightarrow$
  $i_2 = \underset{i}{\arg\max}(g_i^*); j_2 = \underset{j}{\arg\min}(g_j^*)$
  $(u_{i2}^* = +1, u_{j2}^* = -1)$
- case 3: $(i_3, j_3) = \underset{i,j}{\arg\max}\left(g_i + g_j^*\right) \ \Leftrightarrow$
  $i_3 = \underset{i}{\arg\max}(g_i); j_3 = \underset{j}{\arg\max}(g_j^*)$
  $(u_{i3} = +1, u_{j3}^* = +1)$
- case 4: $(i_4, j_4) = \underset{i,j}{\arg\max}\left(-g_i - g_j^*\right) \ \Leftrightarrow$
  $i_4 = \underset{i}{\arg\min}(g_i); j_4 = \underset{j}{\arg\min}(g_j^*)$
  $(u_{i4} = -1, u_{j4}^* = -1)$

Before any modification of α and/or α^*, we must verify that the selected direction is feasible. A point $\alpha^{(*)}$ is feasible if it satisfies all the constraints. A direction vector $u^{(*)}$ is feasible, if there exists $s > 0$ such that $\alpha^{(*)} + su^{(*)}$ is feasible. We must now take into account the inequality constraints. A direction will be feasible if the corresponding Lagrangian multipliers are not still located on the boundaries defined by the inequality constraints, so the search for the optimum directions defined previously becomes:

- case 1: $i_1 = \underset{i:\alpha_i < C}{\arg\max}(g_i)$; $j_1 = \underset{j:\alpha_j > 0}{\arg\min}(g_j)$

- case 2: $i_2 = \underset{i:\alpha_i^*<C}{\arg\max}(g_i^*);\ j_2 = \underset{j:\alpha_j^*>0}{\arg\min}(g_j^*)$
- case 3: $i_3 = \underset{i:\alpha_i<C}{\arg\max}(g_i);\ j_3 = \underset{j:\alpha_j^*<C}{\arg\max}(g_j^*)$
- case 4: $i_4 = \underset{i:\alpha_i>0}{\arg\min}(g_i);\ j_4 = \underset{j:\alpha_j^*>0}{\arg\min}(g_j^*)$

3.2. Determination of the optimal step-size

Considering the quadratic nature of the objective function in (4), $W(\alpha^{(*)})$, it is actually easy to determine, for every case considered previously, the optimal step-size of the gradient algorithm. Given a direction $u^{(*)}$, a feasible point $\alpha^{(*)}$ and in the unconstrained case, the optimal step-size λ_{opt} is defined by: $\lambda_{opt} = \underset{\lambda}{\arg\max}\left(W(\alpha^{(*)} + \lambda u^{(*)})\right)$

$$\lambda_{opt} = \frac{{u^{(*)}}^T \nabla W}{u^{(*)T} K u^{(*)}} \tag{6}$$

where K denotes the Gram matrix (that is not used, see below) and ∇W is the gradient of $W(\alpha^{(*)})$. The optimal step-size of equation (6) can be written as:

- case 1: $\lambda_{opt_1} = \frac{g_{i_1} - g_{j_1}}{k_{i_1 i_1} + k_{j_1 j_1} - 2k_{i_1 j_1}}$
- case 2: $\lambda_{opt_2} = \frac{g_{i_2}^* - g_{j_2}^*}{k_{i_2 i_2} + k_{j_2 j_2} - 2k_{i_2 j_2}}$
- case 3: $\lambda_{opt_3} = \frac{g_{i_3} + g_{j_3}^*}{k_{i_3 i_3} + k_{j_3 j_3} - 2k_{i_3 j_3}}$
- case 4: $\lambda_{opt_4} = \frac{-g_{i_4} - g_{j_4}^*}{k_{i_4 i_4} + k_{j_4 j_4} - 2k_{i_4 j_4}}$

As we can see from the calculation, we are only working in dimension two. Now we have to verify that, after any modification, the updated point remains feasible. An analysis of the first case leads to:

$$\begin{cases} \alpha_{i_1} + \lambda_{opt_1} \leq C \\ \alpha_{j_1} - \lambda_{opt_1} \geq 0 \end{cases} \Rightarrow \lambda_{opt_1} \leq \min(C - \alpha_{i_1}, \alpha_{j_1})$$

If $\lambda_{opt_1} > \min(C - \alpha_{i_1}, \alpha_{j_1})$ the retained value is equal to $\min(C - \alpha_{i_1}, \alpha_{j_1})$. A similar analysis for the other cases gives:

- case 2: $\lambda_{opt_2} \leq \min(C - \alpha_{i_2^*}, \alpha_{j_2^*})$
  else $\lambda_{opt_2} = \min(C - \alpha_{i_2^*}, \alpha_{j_2^*})$
- case 3: $\lambda_{opt_3} \leq \min(C - \alpha_{i_3}, C - \alpha_{j_3^*})$
  else $\lambda_{opt_3} = \min(C - \alpha_{i_3}, C - \alpha_{j_3^*})$
- case 4: $\lambda_{opt_4} \leq \min(\alpha_{i_4}, \alpha_{j_4^*})$
  else $\lambda_{opt_4} = \min(\alpha_{i_4}, \alpha_{j_4^*})$

3.3. Selection of the optimal case

We have now to select the best case in some sense. In our heuristic, that has been experimentally shown efficient, we select the case corresponding to the greatest increase of the objective function $W(\alpha^{(*)})$. Considering again the quadratic nature of $W(\alpha^{(*)})$, and after a few calculations, this increment is given by:

- case 1: $\begin{array}{l} \Delta W_1 = -\lambda_{opt_1} \sum_{k=1}^{N} (k_{i_1 k} - k_{j_1 k})(\alpha_k - \alpha_k^*) \\ -\frac{1}{2}\lambda_{opt_1}^2 (k_{i_1 i_1} + k_{j_1 j_1} - 2k_{i_1 j_1}) \\ +\lambda_{opt_1}(y_{i_1} - y_{j_1}) \end{array}$
- case 2: $\begin{array}{l} \Delta W_2 = -\lambda_{opt_2} \sum_{k=1}^{N} (k_{i_2 k} - k_{j_2 k})(\alpha_k - \alpha_k^*) \\ -\frac{1}{2}\lambda_{opt_1}^2 (k_{i_2 i_2} + k_{j_2 j_2} - 2k_{i_2 j_2}) \\ +\lambda_{opt_1}(y_{i_2} - y_{j_2}) \end{array}$
- case 3: $\begin{array}{l} \Delta W_3 = -\lambda_{opt_3} \sum_{k=1}^{N} (k_{i_3 k} - k_{j_3 k})(\alpha_k - \alpha_k^*) \\ -\frac{1}{2}\lambda_{opt_3}^2 (k_{i_3 i_3} + k_{j_3 j_3} - 2k_{i_3 j_3}) \\ +\lambda_{opt_3}(y_{i_3} - y_{j_3}) - 2\varepsilon\lambda_{opt_3} \end{array}$
- case 4: $\begin{array}{l} \Delta W_4 = -\lambda_{opt_4} \sum_{k=1}^{N} (k_{i_4 k} - k_{j_4 k})(\alpha_k - \alpha_k^*) \\ -\frac{1}{2}\lambda_{opt_4}^2 (k_{i_4 i_4} + k_{j_4 j_4} - 2k_{i_4 j_4}) \\ -\lambda_{opt_4}(y_{i_4} - y_{j_4}) + 2\varepsilon\lambda_{opt_4} \end{array}$

The accepted case is the one that maximizes the increment of the objective function. We need to update the gradient of the objective function. The component $l, (l = 1, \ldots, N)$ of the gradient is updated according to:

- case 1: $\begin{array}{l} \Delta g_l = -\lambda_{opt_1}(k_{i_1 l} - k_{j_1 l}) \\ \Delta g_l^* = \lambda_{opt_1}(k_{i_1 l} - k_{j_1 l}) \end{array}$
- case 2: $\begin{array}{l} \Delta g_l = \lambda_{opt_2}(k_{i_2 l} - k_{j_2 l}) \\ \Delta g_l^* = -\lambda_{opt_2}(k_{i_2 l} - k_{j_2 l}) \end{array}$
- case 3: $\begin{array}{l} \Delta g_l = -\lambda_{opt_3}(k_{i_3 l} - k_{j_3 l}) \\ \Delta g_l^* = \lambda_{opt_3}(k_{i_3 l} - k_{j_3 l}) \end{array}$
- case 4: $\begin{array}{l} \Delta g_l = \lambda_{opt_4}(k_{i_4 l} - k_{j_4 l}) \\ \Delta g_l^* = -\lambda_{opt_4}(k_{i_4 l} - k_{j_4 l}) \end{array}$

3.4. Initialization

The gradient of the objective function is given by:

$$\begin{array}{l} g = \frac{\partial W(\alpha^{(*)})}{\partial \alpha} = -K(\alpha - \alpha^*) - \varepsilon 1_N + y \\ g^* = \frac{\partial W(\alpha^{(*)})}{\partial \alpha^*} = K(\alpha - \alpha^*) - \varepsilon 1_N - y \end{array} \tag{7}$$

At the initialization we set the Lagrange multipliers $\alpha^{(*)}$ to zero (feasible solution), then:

$$\begin{array}{l} g = -\varepsilon 1_N + y \\ \qquad\qquad \text{and} \quad W(0) = 0 \\ g^* = -\varepsilon 1_N - y \end{array} \tag{8}$$

3.5. Algorithm

F-SVR learning algorithm is resumed as follows:

1. Initialization.
2. Evaluation of the best direction, optimal step-size and increment of the objective function for each case.
3. Selection of the case that maximizes this increment.
4. Update of the solution, the gradient and the objective function.
5. Test for convergence, if not, return to step 2.

4. EXPERIMENTAL RESULTS

We compare now our F-SVR algorithm with an implementation of SVMLight for regression problems, only for learning results comparison (no test data were used here). For SVM-Light, we consider different sizes of the working set. The first dataset is generated according to a noisy nonlinear model (see Fig 1). For this first experiment, both algorithms are written in MATLAB. Data are composed of 2000 observations of one predictor and the corresponding response. For this dataset, $\varepsilon = 0.2$, $C = 10$ and the gaussian kernel parameter σ is equal to 0.3. Results are summarized in Table 1 where we evaluate the training Normalised Mean Square Error NMSE[1].

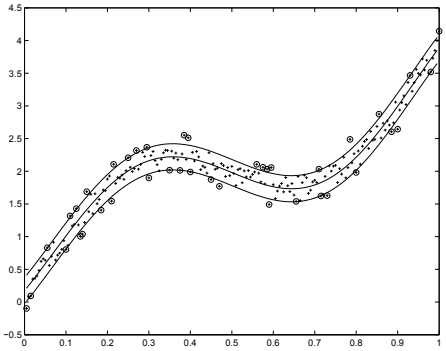

Fig. 1. Simulated data (200 observations shown), SV are indicated by a circle

The second dataset (Census-house[2]) is concerned with predicting the median price of a house using some demographic information. Data are composed of 20000 observations (with 121 attributes). We compare our F-SVR algorithm to LIBSVM and SVMLIGHT[3] for regression problems. All algorithms are here written in C. For this experiment $\varepsilon = 10^{-6}$, $C = 1000$ and $\sigma = 100$. Obtained results are shown in Table 2.

[1] NMSE $= \frac{\sum_{i=1}^{N}(f(x_i)-y_i)^2}{\sum_{i=1}^{N}(y_i)^2}$

[2] http://www.cs.toronto.edu/ delve/data/census-house/

[3] http://www.csie.ntu.edu.tw/ cjlin/libsvm/, http://svmlight.joachims.org/

	Time (sec)	Working set	NMSE
F-SVR algorithm	26.672	2	0.0048
SVMLight	199.125	2	0.024
	218.750	4	-
	253.016	8	-
	397.172	16	-
	722.328	32	-

Table 1. Results comparison for simulated data

	Time (sec)	Working set	NMSE
F-SVR algorithm	3742.99	2	0.0019
SVMLIGHT	5044.36	2	0.019
LIBSVM	5708.61	-	0.019

Table 2. Results comparison for Census-housing data

5. CONCLUSION

In this paper, we introduced a new learning algorithm for regression problems. This algorithm is easy to implement and does not require the storage of the Gram matrix. It efficiently uses the nature of the constraints of the optimization problem and properties of quadratic functions. We showed that this algorithm is fast and efficient for training large data sets, as proved our experiments. Development of an online version of our algorithm is under investigation.

6. REFERENCES

[1] Smola A.J. and Scholkopf B., "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, no. 3, pp. 199–222, 2004.

[2] Platt J.C., "Fast training of support vector machines using sequential minimal optimization," *Advances in Kernel Methods: Support Vector Learning*, pp. 185–208, 1999.

[3] S. Weston J. Bottou L. Bordes A., Ertekin, "Fast kernel classifiers with online and active learning.," *Journal of Machine Learning Research (JMLR),*, vol. 6, pp. 1579–1619, 2005.

[4] Bengio S. Collobert R., "Svmtorch: Support vector machines for large-scale regression problems," *Journal of Machine Learning Research*, , no. 1, pp. 143–160, 2001.

[5] Joachims T., "Making large-scale support vector machine learning practical," *Advances in Kernel Methods: Support Vector Machines*, pp. 169–184, 1999.

[6] Vapnik V., *The nature of statistical learning theory*, Springer, New York, 1995.