Weighted Least Squares Support Vector Machines: robustness and sparse approximation *

J.A.K. Suykens, J. De Brabanter, L. Lukas, and J. Vandewalle

Katholieke Universiteit Leuven

Department of Electrical Engineering, ESAT-SISTA

Kardinaal Mercierlaan 94, B-3001 Leuven (Heverlee), Belgium

Tel: 32/16/32 18 02 Fax: 32/16/32 19 70

Email: johan.suykens@esat.kuleuven.ac.be

Corresponding author: Johan Suykens

To appear in
Neurocomputing special issue

*This research work was carried out at the ESAT laboratory and the Interdisciplinary Center of Neural Networks ICNN of the Katholieke Universiteit Leuven, in the framework of the FWO project *Learning and Optimization: an Interdisciplinary Approach*, the Belgian Program on Interuniversity Poles of Attraction, initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture (IUAP P4-02 & IUAP P4-24) and the Concerted Action Project MEFISTO of the Flemish Community. Johan Suykens is a postdoctoral researcher with the National Fund for Scientific Research FWO - Flanders.

Abstract

Least Squares Support Vector Machines (LS-SVM) is an SVM version which involves equality instead of inequality constraints and works with a least squares cost function. In this way the solution follows from a linear Karush-Kuhn-Tucker system instead of a quadratic programming problem. However, sparseness is lost in the LS-SVM case and the estimation of the support values is only optimal in the case of a Gaussian distribution of the error variables. In this paper we discuss a method which can overcome these two drawbacks. We show how to obtain robust estimates for regression by applying a weighted version of LS-SVM. We also discuss a sparse approximation procedure for weighted and unweighted LS-SVM. It is basically a pruning method which is able to do pruning based upon the physical meaning of the sorted support values, while pruning procedures for classical multilayer perceptrons require the computation of a Hessian matrix or its inverse. The methods of this paper are illustrated for RBF kernels and demonstrate how to obtain robust estimates with selection of an appropriate number of hidden units, in the case of outliers or non-Gaussian error distributions with heavy tails.

Keywords. Support Vector Machines, (Weighted) Least Squares, Ridge Regression, Sparse Approximation, Robust Estimation.

1 Introduction

Support Vector Machines (SVM) for classification and nonlinear function estimation, as introduced by Vapnik [35, 36] and further investigated by many others [4, 21, 22, 23, 24, 25], is an important new methodology in the area of neural networks and nonlinear modelling [27]. While classical neural networks approaches, such as multilayer perceptrons (MLP) and radial basis function (RBF) networks [2, 17], suffer from problems like the existence of many local minima and the choice of the number of hidden units [2, 13], SVM solutions are characterized by convex optimization problems, up to the determination of a few additional tuning parameters. Moreover, also the model complexity follows from this convex optimization problem. Typically, one solves a convex quadratic programming (QP) problem in dual space in order to determine the SVM model. The formulation of the optimization problem in the primal space associated with this QP problem involves inequality constraints. In the case of function estimation it is related to Vapnik's epsilon-insensitive loss function. An interesting property of the SVM solution is that one obtains a sparse approximation, in the sense that many elements in the QP solution vector are equal to zero. The additional hyperparameters of the SVM model are often determined by model selection based upon generalization bounds, which have been derived within the area of statistical learning theory. SVM is a kernel based approach, which allows the use of linear, polynomial and RBF kernels and others that satisfy Mercer's condition.

Recently, least squares (LS) versions of SVM's have been investigated for classification [28] and function estimation [20]. In these LS-SVM formulations one works with equality instead of inequality constraints and a sum squared error (SSE) cost function as it is frequently used in training of classical neural networks. This reformulation greatly simplifies the problem in such a way that the solution is characterized by a linear system, more precisely a KKT (Karush-Kuhn-Tucker) system [8], which takes a similar form as the linear system that one solves in every iteration step by interior point methods for standard SVM's [26]. This linear system can be efficiently solved by iterative methods such as conjugate gradient [29]. However, despite these computationally attractive features, LS-SVM solutions also have some potential drawbacks. A first drawback is that sparseness is lost in the LS-SVM solution. In this case every data point is contributing to the model and the relative importance of a data point is given by its support value. A second drawback is that it is well-known that the use of a SSE cost function without regularization might lead to estimates which are less robust, e.g. with respect to outliers on the data or when the underlying assumption of a Gaussian distribution for the error variables is not realistic.

The aim of this paper is to show that one can overcome these drawbacks concerning sparseness and robustness within the present LS-SVM framework. First of all one should note that only the output weights of the SVM model follow as solution to the linear system. Only these parameters are related to the SSE cost function which means that (up to a certain extent) one can still correct for wrong assumptions by an appropriate choice of the hyperparameters. These can be determined in several possible ways such as cross-validation, bootstrapping, VC bounds, Bayesian inference etc. [4, 34]. We show in this paper how one can apply weighted least squares in order to produce a more robust estimate. This is done by first applying an (unweighted) LS-SVM and, in a second stage, associate weighting values to the error variables based upon the resulting error variables from the first stage. Techniques of weighted least squares are well-known e.g. in statistics, identification and control theory and signal processing. For LS-SVM's it can be employed as a cheap and efficient way to robustify the solution. In this way it can be used as an alternative to other methods in robust estimation [14] as L_1 estimators and M-estimators with Huber loss function. Robust estimation is also possible within the standard SVM context [36]. In standard SVM methodology one chooses a given cost function (any convex cost function can be taken in principle as shown in [26]). Instead of this top-down approach the weighted LS-SVM approach aims at working bottom-up by solving a sequence of weighted LS-SVM's starting from the unweighted version. In this way one implicitly tries to find an optimal underlying cost function, instead of imposing the cost function beforehand. In this sense there is also a close link between solving the SVM problem for a given convex cost function by interior point methods and iterative weighting of LS-SVM solutions.

Furthermore, in this paper we illustrate how sparseness can be imposed to the weighted LS-SVM solution by gradually pruning the sorted support value spectrum. While in pruning methods for MLP's [2] (like optimal brain damage [16] and optimal brain surgeon [12]) the procedure involves a Hessian matrix or its inverse, the pruning in LS-SVM's can be done based upon the solution vector itself (note that this implicitly requires inverting the system). Less meaningful data points as indicated by their support values, are removed and the LS-SVM is re-computed on the basis of the remaining points while validating on the complete training data set. In this paper we focus on a cheap and simple pruning method. Other methods for obtaining a sparse approximation with LS-SVM's are possible [6]. The advantage of the method shown in this paper is that the determination of the hyperparameters can be kept localized, while in the other approaches one needs to solve the convex optimization problem (which implicitly corresponds to solving a sequence of linear systems) for a given set of hyperparameters.

In general, parametric models like MLP's or RBF networks are applicable within a broad range of applications of either static problems (classification, regression, density estimation) or dynamic problems (recurrent networks, optimal control). Standard SVM's on the other hand have only been applied to static problems. However, the use of equality constrained and SSE based formulations within LS-SVM greatly simplifies the formulations and allows us to extend the method to recurrent networks [31] and control applications [32]. In the latter cases convexity is lost, but some of the other interesting SVM properties remain applicable. The results of this paper on weighted LS-SVM's for robust estimation and sparse approximation further motivates the LS-SVM approach towards its use as a new and general neural network methodology.

This paper is organized as follows. In Section 2 we review some basic notions of LS-SVM's for function estimation. In Section 3 we discuss the weighted LS-SVM formulation. The sparse approximation procedure is discussed in Section 4. Finally, illustrative examples are given in Section 5.

2 LS-SVM for Nonlinear Function Estimation

Given a training data set of N points $\{x_k, y_k\}_{k=1}^{N}$ with input data $x_k \in \mathbb{R}^n$ and output data $y_k \in \mathbb{R}$, one considers the following optimization problem in primal weight space:

$$\min_{w,b,e} J(w,e) = \frac{1}{2} w^T w + \frac{1}{2} \gamma \sum_{k=1}^{N} e_k^2 \tag{1}$$

such that

$$y_k = w^T \varphi(x_k) + b + e_k \, , \quad k = 1, ..., N$$

with $\varphi(\cdot) : \mathbb{R}^n \rightarrow \mathbb{R}^{n_h}$ a function which maps the input space into a so-called higher dimensional (possibly infinite dimensional) feature space, weight vector $w \in \mathbb{R}^{n_h}$ in primal weight space, error variables $e_k \in \mathbb{R}$ and bias term b. Note that the cost function J consists of a SSE fitting error and a regularization term, which is also a standard procedure for the training of MLP's and is related to ridge regression [10]. The relative importance of these terms is determined by the positive real constant γ. In the case of noisy data one avoids overfitting by taking a smaller γ value. SVM problem formulations of this form have been investigated independently in [20] (without bias term) and [28].

In primal weight space one has the model

$$y(x) = w^T \varphi(x) + b. \tag{2}$$

The weight vector w can be infinite dimensional, which makes a calculation of w from (1) impossible in general. Therefore, one computes the model in the dual space instead of the primal space. One defines the Lagrangian

$$\mathcal{L}(w, b, e; \alpha) = J(w, e) - \sum_{k=1}^{N} \alpha_k \left\{ w^T \varphi(x_k) + b + e_k - y_k \right\} \tag{3}$$

with Lagrange multipliers $\alpha_k \in \mathbb{R}$ (called support values). The conditions for optimality are given by

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial w} = 0 & \rightarrow \quad w = \sum_{k=1}^{N} \alpha_k \varphi(x_k) \\ \frac{\partial \mathcal{L}}{\partial b} = 0 & \rightarrow \quad \sum_{k=1}^{N} \alpha_k = 0 \\ \frac{\partial \mathcal{L}}{\partial e_k} = 0 & \rightarrow \quad \alpha_k = \gamma e_k, \qquad\qquad\qquad\quad k = 1, ..., N \\ \frac{\partial \mathcal{L}}{\partial \alpha_k} = 0 & \rightarrow \quad w^T \varphi(x_k) + b + e_k - y_k = 0, \quad k = 1, ..., N. \end{cases} \tag{4}$$

These conditions are similar to standard SVM optimality conditions, except for the condition $\alpha_k = \gamma e_k$. At this point one looses the sparseness property in LS-SVM's [9].

After elimination of w, e one obtains the solution

$$\left[\begin{array}{c|c} 0 & 1_v^T \\ \hline 1_v & \Omega + \frac{1}{\gamma} I \end{array} \right] \left[\begin{array}{c} b \\ \hline \alpha \end{array} \right] = \left[\begin{array}{c} 0 \\ \hline y \end{array} \right] \tag{5}$$

with $y = [y_1; ...; y_N]$, $1_v = [1; ...; 1]$, $\alpha = [\alpha_1; ...; \alpha_N]$ and $\Omega_{kl} = \varphi(x_k)^T \varphi(x_l)$ for $k, l = 1, ..., N$. According to Mercer's condition, there exists a mapping φ and an expansion

$$K(x, y) = \sum_i \varphi_i(x) \varphi_i(y), \qquad x, y \in \mathbb{R}^n, \tag{6}$$

if and only if, for any $g(x)$ such that $\int g(x)^2 dx$ is finite, one has

$$\int K(x,y)\, g(x)g(y)\, dxdy \geq 0. \tag{7}$$

As a result, one can choose a kernel $K(\cdot,\cdot)$ such that

$$K(x_k, x_l) = \varphi(x_k)^T \varphi(x_l), \qquad k,l = 1, ..., N \tag{8}$$

The resulting LS-SVM model for function estimation becomes

$$y(x) = \sum_{k=1}^{N} \alpha_k K(x, x_k) + b \tag{9}$$

where α, b are the solution to (5). We focus on the choice of an RBF kernel $K(x_k, x_l) = \exp\{-\|x_k - x_l\|_2^2/\sigma^2\}$ for the sequel.

When solving large linear systems, it is necessary to apply iterative methods [10] to (5). However, the matrix in (5) is not positive definite. According to [29] one can transform the system into a positive definite system such that iterative methods (conjugate gradient, successive overrelaxation or others) can be applied to it. Note that the computational complexity of the conjugate gradient method for solving a linear system $\mathcal{A}x = \mathcal{B}$ is $\mathcal{O}(Nr^2)$ where $\mathrm{rank}(\mathcal{C}) = r$ with $\mathcal{A} = I + \mathcal{C}$ and $\mathcal{A} \in \mathbb{R}^{N\times N}$. It converges in at most $r+1$ steps. The speed of convergence depends on the condition number of the matrix. In the case of LS-SVM this is influenced by the choice of (γ, σ) when using an RBF kernel.

3 Robust Estimation by Weighted LS-SVM

In order to obtain a robust estimate based upon the previous LS-SVM solution, in a subsequent step, one can weight the error variables $e_k = \alpha_k/\gamma$ by weighting factors v_k. This leads to the optimization problem:

$$\min_{w^\star, b^\star, e^\star} J(w^\star, e^\star) = \frac{1}{2} w^{\star T} w^\star + \frac{1}{2}\gamma \sum_{k=1}^{N} v_k\, e_k^{\star 2} \tag{10}$$

such that

$$y_k = w^{\star T}\varphi(x_k) + b^\star + e_k^\star, \qquad k = 1, ..., N.$$

The Lagrangian becomes

$$\mathcal{L}(w^\star, b^\star, e^\star; \alpha^\star) = J(w^\star, e^\star) - \sum_{k=1}^{N} \alpha_k^\star \{w^{\star T}\varphi(x_k) + b^\star + e_k^\star - y_k\}. \tag{11}$$

The unknown variables for this weighted LS-SVM problem are denoted by the $\star$ symbol. From the conditions for optimality and elimination of $w^\star, e^\star$ one obtains the KKT system:

$$\left[\begin{array}{c|c} 0 & 1_v^T \\ \hline 1_v & \Omega + V_\gamma \end{array}\right] \left[\begin{array}{c} b^\star \\ \hline \alpha^\star \end{array}\right] = \left[\begin{array}{c} 0 \\ \hline y \end{array}\right] \tag{12}$$

where the diagonal matrix V_γ is given by

$$V_\gamma = \text{diag}\{\frac{1}{\gamma v_1}, ..., \frac{1}{\gamma v_N}\}. \tag{13}$$

The choice of the weights v_k is determined based upon the error variables $e_k = \alpha_k/\gamma$ from the (unweighted) LS-SVM case (5). Robust estimates are obtained then (see [5, 18]) e.g. by taking

$$v_k = \begin{cases} 1 & \text{if} \quad |e_k/\hat{s}| \leq c_1 \\ \dfrac{c_2 - |e_k/\hat{s}|}{c_2 - c_1} & \text{if} \quad c_1 \leq |e_k/\hat{s}| \leq c_2 \\ 10^{-4} & \quad \text{otherwise} \end{cases} \tag{14}$$

where $\hat{s}$ is a robust estimate of the standard deviation of the LS-SVM error variables e_k:

$$\hat{s} = \frac{\text{IQR}}{2 \times 0.6745}. \tag{15}$$

The interquartile range IQR is the difference between the 75th percentile and 25th percentile. In the estimate $\hat{s}$ one takes into account how much the estimated error distribution deviates from a Gaussian distribution. Another robust estimate of the standard deviation is $\hat{s} = 1.483\,\text{MAD}(x_i)$ where MAD stands for the median absolute deviation [11]. The SSE cost function in the unweighted LS-SVM formulation is optimal under the assumption of a normal Gaussian distribution for e_k. The procedure (14) corrects for this assumption in order to obtain a robust estimate when this distribution is not normal. Eventually, the procedure (10)(14) can be repeated iteratively, but in practice one single additional weighted LS-SVM step will often be sufficient. One assumes that e_k has a symmetric distribution which is usually the case when (γ, σ) are well-determined by an appropriate model selection method. The constants c_1, c_2 are typically chosen as $c_1 = 2.5$ and $c_2 = 3$ [18]. This is a reasonable choice taking into account the fact that for a Gaussian distribution there will be very few residuals larger than $2.5\hat{s}$. Another possibility is to determine c_1, c_2

from a density estimation of the e_k distribution. Using these weightings one can correct for y-outliers or for a non-Gaussian instead of Gaussian error distributions.

This leads us to the following algorithm:

Algorithm 1 - Weighted LS-SVM

1. *Given training data* $\{x_k, y_k\}_{k=1}^N$, *find an optimal* (γ, σ) *combination (e.g. by 10-fold cross-validation or generalization bounds) by solving linear systems (5). For the optimal* (γ, σ) *combination one computes* $e_k = \alpha_k/\gamma$ *from (5).*

2. *Compute* $\hat{s}$ *from the* e_k *distribution.*

3. *Determine the weights* v_k *based upon* e_k, $\hat{s}$.

4. *Solve the Weighted LS-SVM (12), giving the model* $y(x) = \sum_{k=1}^N \alpha_k^\star K(x, x_k) + b^\star$.

An important notion in robust estimation is the breakdown point of an estimator [1, 6, 19]. Loosely speaking it is the smallest fraction of contamination of a given data set that can result in an estimate which is arbitrarily far away from the estimated parameter vector obtained from the uncontaminated data set. It is well known that the least squares estimate in linear regression (parametric) without regularization has a low breakdown point. At this point the unweighted LS-SVM has already better properties due to the fact that least squares is only applied to finding α (which corresponds to estimating the output layer and not (γ, σ)). It is still possible then to correct with (γ, σ) when using an RBF kernel. Although unweighted LS-SVM already has rather desirable properties, the breakdown point is further improved by applying the weighted LS-SVM afterwards.

In standard SVM approaches one usually chooses a certain cost function (any convex cost function can be taken according to [26]). However, in such a top-down approach one assumes in fact that the noise distribution is known because one knows which cost function is optimal for a given noise distribution. The weighted LS-SVM procedure tries to work bottom-up and aims at finding the optimal underlying cost function for the given noisy data. The methods can be conceptually linked by the fact that when one applies an interior point method for solving a standard SVM problem, at every iteration step a KKT system is solved which is similar to solving one single LS-SVM. Hence, solving a standard SVM with arbitrary convex cost function implicitly corresponds in fact to solving a sequence of LS-SVMs.

4 Imposing Sparseness

While standard SVM's possess a sparseness property in the sense that many α_k values are equal to zero, this is not the case for LS-SVM's due to the fact that $\alpha_k = \gamma e_k$ from the conditions for optimality. An equivalence has also been proven between sparse approximation and SVM's [9]. Here we discuss a simple procedure how to obtain a sparse approximation for LS-SVM's. This is done by exploiting the fact that the support values have a physical meaning in the sense that they reveal the relative importance of the data points for contributing to the model.

We propose here a pruning procedure for LS-SVM's which is based upon the sorted support value spectrum. By omitting a relative and small amount of the least meaningful data points (this corresponds to setting these α_k values to zero) and re-estimating the LS-SVM one obtains a sparse approximation. In order to guarantee a good generalization performance, in each of these pruning steps one can optimize (γ, σ) (by defining an independent validation set, 10-fold cross-validation, generalization bounds or others). An important difference with pruning methods for MLP's [2, 12, 16], e.g. optimal brain damage and optimal brain surgeon, is that pruning can be done based upon solution vector itself and doesn't require the knowledge of a Hessian matrix or its inverse. This pruning method has been successfully applied to unweighted LS-SVM's [30]. Here we outline the algorithm for the weighted LS-SVM case.

Algorithm 2 - Weighted LS-SVM Pruning

1. *Set $N = N_{tot}$ equal to the number of training data.*

2. *Given N training data, apply Algorithm 1 where (γ, σ) is determined on the total amount of N_{tot} training data. The solution to the linear system for optimal (γ, σ) yields $\alpha_k^\star$.*

3. *Sort the values $|\alpha_k^\star|$.*

4. *Remove a small amount of M points (typically 5% of the N points) that have the smallest values in the sorted $|\alpha_k^\star|$ spectrum.*

5. *Retain $N - M$ points and set $N := N - M$.*

6. *Go to 2 and retrain on the reduced training set, unless the user-defined performance index degrades.*

Typically, doing a number of pruning steps without modification of (γ, σ) will be possible. When the generalization performance starts degrading (checked e.g. on a validation set or by means of cross-validation as illustrated in [33]) an update of (γ, σ) will be needed. The fact that this (γ, σ) determination can be kept 'localized' is a possible advantage of this method in comparison with other approaches which need to solve a QP problem for the several possible choices of the hyperparameters. Also note that the pruning method is in fact an iterative scheme where in each step one has to solve a KKT system. For interior point methods in solving standard quadratic programming type SVM's [26], at each iteration step one solves a KKT system which has a similar form as in the LS-SVM case. However, in Algorithm 2 the size of the linear system that one solves decreases from step to step because N decreases. The convergence of Algorithm 2 for obtaining sparseness is guaranteed by construction.

5 Examples

In this example we illustrate the method of weighted LS-SVM and sparse approximation. First, we show two examples of estimating a sinc function from noisy data: (a) strong outliers are superimposed on zero mean Gaussian noise distribution (Fig.1-2); (b) non-Gaussian noise distribution (central t-distribution with 4 degrees of freedom, i.e. heavy tails [15]) (Fig.4-5). Given is a training set of $N = 300$ data points.

From the simulation results it is clear that in both cases the unweighted LS-SVM is quite robust and does not breakdown (Fig.1) (Fig.4). The generalization performance is further improved by applying weighted LS-SVM (Algorithm 1), shown in Fig.2 and Fig.5, respectively. The (γ, σ) hyperparameters are determined here by means of 10-fold cross-validation on the training data. The good generalization performance on fresh test data is shown for all cases. Fig.2 and Fig.5 show that the use of weighted LS-SVM leads to a e_k distribution which is closer to a Gaussian distribution, which leads to better estimates within the LS-SVM context.

An additional comparison with a standard SVM with Vapnik ϵ-insensitive loss function is made. The Matlab SVM Toolbox by Steve Gunn was used to generate the SVM results. Here $\epsilon = 0$ was taken and as upper bound on support values $C = Inf$. An optimal σ value was selected. Other ϵ, C combinations resulted in worse results. These comparative results are shown in (Fig.3) (Fig.6). In these examples the weighted LS-SVM results show the best results. The unweighted LS-SVM is also quite robust. Due to the choice of a 2-norm this may sound surprising. However, one should be aware that only the output weights (support values α) follow from the solution to the linear system while (γ, σ) are to be determined at another level.

Fig.7-8 shows comparative results on the motorcycle data, a well-known benchmark data set in statistics [7]. The x values are time measurements in milliseconds after simulated impact and the y values are measurements of head acceleration. The x values are not equidistant and in some cases multiple y observations are present for certain x values. The data are heteroscedastic. In this sense it forms a challenging test case. Fig.7-8 show the results from unweighted and weighted LS-SVM in comparison with standard SVM. In this example standard SVM suffers more from boundary effects. The tuning parameters are: $\gamma = 2$, $\sigma = 6.6$ (LS-SVM) and $\sigma = 11$, $\epsilon = 0$, $C = Inf$ (Vapnik SVM).

Fig.9 shows the improvements of weighted LS-SVM on the Boston housing data in comparison with unweighted LS-SVM. The weighted LS-SVM achieves an improved test set performance after determination of (γ, σ) by 10-fold CV on a randomly selected training set of 406 points. The remaining test set consisted of 100 points. The data were normalized except the binary variables. Optimal values of (γ, σ) were determined by 10-fold CV on the training set. The weighted LS-SVM resulted in a test set MSE error of 0.1638, which was an improvement over the unweighted LS-SVM test set MSE error of 0.1880. The improved performance is achieved by suppressing the outliers in the histogram shown in Fig.9.

The sparse approximation procedure (Algorithm 2) is illustrated on Fig.10-11. These results are obtained by starting from the training data of Fig.1-2. The Figures show that robust estimates are obtained by a combination of weighted LS-SVM and sparse approximation by applying Algorithm 2. Some shiftings in the sorted support value spectrum are shown in Fig.11. In this example about 20 support vectors are needed to guarantee a good generalization performance. As shown in [33] on many UCI data sets, optimizing the hyperparameters during the pruning process will improve the generalization performance. We want to stress here that pruning of the LS-SVM is in fact not needed, unless one wants to obtain a sparse representation of the original model.

6 Conclusions

We have shown how to obtain robust estimates within the LS-SVM framework in the case of outliers and heavy tailed non-Gaussian error distributions. This is done here by applying a weighted LS-SVM version. While least squares in standard parametric linear regression has a low breakdown point, LS-SVM's with RBF kernel have much better properties. Nevertheless, the robustness can be further enhanced by applying an additional weighted LS-SVM step. While standard SVM's possess a sparseness property for the solution vector, this feature is lost when considering LS-SVM's. However, we have shown how sparseness can be obtained by pruning the sorted support value spectrum if needed. The pruning procedure is based upon the solution vector itself which is an advantage in comparison with classical multilayer perceptron pruning. This procedure has the potential advantage of keeping the hyperparameter selection more localized. While standard SVM approaches start from choosing a given convex cost function and obtain a robust and sparse estimate in a top-down fashion, this procedure has the disadvantage that one should know in fact beforehand which cost function is statistically optimal. In this paper we have successfully demonstrated an alternative bottom-up procedure which starts from an unweighted LS-SVM and then robustifies the solution by defining weightings based upon the error distribution. This motivates further fundamental research in this direction for future work.

References

[1] Andrews D.F., Bichel P.J., Hampel F.R., Huber P.J., Rogers W.H., Tukey J.W., *Robust estimates of location: survey and advances*, Princeton, NJ: Princeton University Press, 1972.

[2] Bishop C.M., *Neural networks for pattern recognition*, Oxford University Press, 1995.

[3] Chen S.C., Donoho D.L., Saunders M.A., "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, Vol.20, No.1, pp.33-61, 1998.

[4] Cristianini N., Shawe-Taylor J., *An Introduction to Support Vector Machines*, Cambridge University Press, 2000.

[5] David H.A., "Early sample measures of variability," *Statistical Science*, Vol.13, No.4, pp.368-377, 1998.

[6] Donoho D.L., Huber P.J., "The notion of breakdown point," in A Festschrift for Erich Lehmann, (Ed. P. Bickel, K. Doksum, J.L. Hodges Jr.), Belmont, CA: Wadsworth, 1983.

[7] Eubank R.L., *Nonparametric regression and spline smoothing*, Statistics: textbooks and monographs, Vol. 157, second edition, Marcel Dekker, New York, 1999.

[8] Fletcher R., *Practical methods of optimization*, New York: Wiley, 1987.

[9] Girosi F., "An equivalence between sparse approximation and support vector machines," *Neural Computation*, 10(6), 1455-1480, 1998.

[10] Golub G.H., Van Loan C.F., *Matrix Computations*, Baltimore MD: Johns Hopkins University Press, 1989.

[11] Hampel F.R., Ronchetti E.M., Rousseeuw P.J., Stahel W.A., *Robust statistics: the approach based on influence functions*, John Wiley & Sons, New York, 1986.

[12] Hassibi B., Stork D.G., "Second order derivatives for network pruning: optimal brain surgeon," In Hanson, Cowan, Giles (Eds.) *Advances in Neural Information Processing Systems*, Vol.5, pp.164-171, San Mateo, CA: Morgan Kaufmann, 1993.

[13] Hornik K., Stinchcombe M., White H., "Multilayer feedforward networks are universal approximators," *Neural Networks*, Vol.2, pp.359-366, 1989.

[14] Huber P.J., *Robust statistics*, New York: Wiley, 1981.

[15] Johnson N.L., Kotz S., *Distributions in Statistics: Continuous Univariate Distributions*, Vol.1-2, New York, Wiley, 1970.

[16] Le Cun Y., Denker J.S., Solla S.A., "Optimal brain damage," In Touretzky (Ed.) *Advances in Neural Information Processing Systems*, Vol.2, pp.598-605, San Mateo, CA: Morgan Kaufmann, 1990.

[17] Poggio T., Girosi F., "Networks for approximation and learning," *Proceedings of the IEEE*, Vol.78, No.9, pp.1481-1497, 1990.

[18] Rousseeuw P.J., Leroy A., *Robust regression and outlier detection*, New York: Wiley, 1987.

[19] Rousseeuw P.J., van Zomeren B.C., "Unmasking multivariate outliers and leverage points," *Journal of the American Statistical Association*, 85, 633-639, 1990.

[20] Saunders C., Gammerman A., Vovk V., "Ridge Regression Learning Algorithm in Dual Variables," *Proc. of the 15th Int. Conf. on Machine Learning (ICML'98)*, Morgan Kaufmann, pp.515-521, 1998.

[21] Schölkopf B., Sung K.-K., Burges C., Girosi F., Niyogi P., Poggio T. & Vapnik V., "Comparing support vector machines with Gaussian kernels to radial basis function classifiers," *IEEE Transactions on Signal Processing,* Vol.45, No.11, pp.2758-2765, 1997.

[22] Schölkopf B., Burges C., Smola A. (Eds.), *Advances in Kernel Methods - Support Vector Learning*, MIT Press, 1998.

[23] Schölkopf B., Mika S., Burges C., Knirsch P., Müller K.-R., Rätsch G., Smola A., "Input Space vs. Feature Space in Kernel-Based Methods," *IEEE Transactions on Neural Networks*, Vol.10, No.5, pp.1000-1017, 1999.

[24] Smola A., Schölkopf B., Müller K.-R., "The connection between regularization operators and support vector kernels," *Neural Networks*, 11, 637-649, 1998.

[25] Smola A., Schölkopf B., "On a kernel-based method for pattern recognition, regression, approximation and operator inversion," *Algorithmica*, 22, 211-231, 1998.

[26] Smola A., *Learning with Kernels*, PhD Thesis, published by: GMD, Birlinghoven, 1999.

[27] Suykens J.A.K., Vandewalle J. (Eds.) *Nonlinear Modeling: advanced black-box techniques*, Kluwer Academic Publishers, Boston, 1998.

[28] Suykens J.A.K., Vandewalle J., "Least squares support vector machine classifiers," *Neural Processing Letters*, Vol.9, No.3, pp.293-300, June 1999.

[29] Suykens J.A.K., Lukas L., Van Dooren P., De Moor B., Vandewalle J., "Least squares support vector machine classifiers: a large scale algorithm," *European Conference on Circuit Theory and Design, (ECCTD'99)*, pp.839-842, Stresa Italy, August 1999.

[30] Suykens J.A.K., Lukas L., Vandewalle J., "Sparse least squares support vector machine classifiers," *European Symposium on Artificial Neural Networks (ESANN 2000)*, Bruges Belgium, pp.37-42, April 2000.

[31] Suykens J.A.K., Vandewalle J., "Recurrent least squares support vector machines," *IEEE Transactions on Circuits and Systems-I*, Vol.47, No.7, pp.1109-1114, Jul. 2000.

[32] Suykens J.A.K., Vandewalle J., De Moor B., "Optimal Control by Least Squares Support Vector Machines," *Neural Networks*, vol. 14, no. 1, pp.23-35, Jan. 2001.

[33] Van Gestel T., Suykens J.A.K., Baesens B., Viaene S., Vanthienen J., Dedene G., De Moor B., Vandewalle J., "Benchmarking Least Squares Support Vector Machine Classifiers," *Internal Report 00-37, ESAT-SISTA, K.U.Leuven.*

[34] Van Gestel T., Suykens J.A.K., Baestaens D., Lambrechts A., Lanckriet G., Vandaele B., De Moor B., Vandewalle J., "Financial time series prediction using least squares support vector machines within the evidence framework," *IEEE Transactions on Neural Networks* (special issue on Neural Networks in Financial Engineering), in press.

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

[36] Vapnik V., *Statistical learning theory,* John Wiley, New-York, 1998.

Captions of Figures

Figure 1: Estimation of a sinc function by LS-SVM with RBF kernel, given 300 training data point, corrupted by zero mean Gaussian noise and 3 outliers (denoted by '+'). (Top-Left) Training data set; (Top-Right) resulting LS-SVM model evaluated on an independent test set: (solid line) true function, (dashed line) LS-SVM estimate; (Bottom-Left) $e_k = \alpha_k/\gamma$ values; (Bottom-Right) histogram for distribution of e_k values, which is non-Gaussian due to the 3 outliers.

Figure 2: (Continued) Weighted LS-SVM applied to the results of Fig.1. The e_k distribution becomes Gaussian and the generalization performance on the test data improves.

Figure 3: (Continued) Comparison between standard SVM, unweighted LS-SVM and weighted LS-SVM. Weighted LS-SVM gives the best estimate for this example.

Figure 4: Estimation of a sinc function by LS-SVM with RBF kernel, given 300 training data point, corrupted by a central t-distribution with heavy tails. (Top-Left) Training data set; (Top-Right) resulting LS-SVM model evaluated on an independent test set: (solid line) true function, (dashed line) LS-SVM estimate; (Bottom-Left) $e_k = \alpha_k/\gamma$ values; (Bottom-Right) histogram for distribution of e_k values, which is non-Gaussian.

Figure 5: (Continued) Weighted LS-SVM applied to the results from Fig.4. The e_k distribution becomes Gaussian and the generalization performance on the test data improves.

Figure 6: (Continued) Comparison between standard SVM, unweighted LS-SVM and weighted LS-SVM. Weighted LS-SVM gives the best estimate for this example.

Figure 7: Motorcycle dataset: comparison between standard SVM, unweighted LS-SVM and weighted LS-SVM.

Figure 8: (Continued) enlarged parts of the previous Figure. From (Top-Left) it is visible that standard SVM suffers more from boundary effects with oscillatory behaviour. From

(Bottom-Left) one observes that weighted LS-SVM improves in this region because of the correction for outliers.

Figure 9: Boston housing data set: (Top) histogram of e_k for unweighted LS-SVM with RBF kernel. The outliers are clearly visible; (Bottom) histogram resulting from weighted LS-SVM with improved test set performance.

Figure 10: Sparse approximation procedure of Algorithm 2, applied to the data of Fig.1-2. Shown are some intermediate pruning results starting from 300 data points: 300 support vectors (SV) (Top-Left); 100 SV (Top-Right); 50 SV (Bottom-Left); 20 SV (Bottom-Right).

Figure 11: Sorted support value spectrum related to the 4 steps shown in the previous Figure.

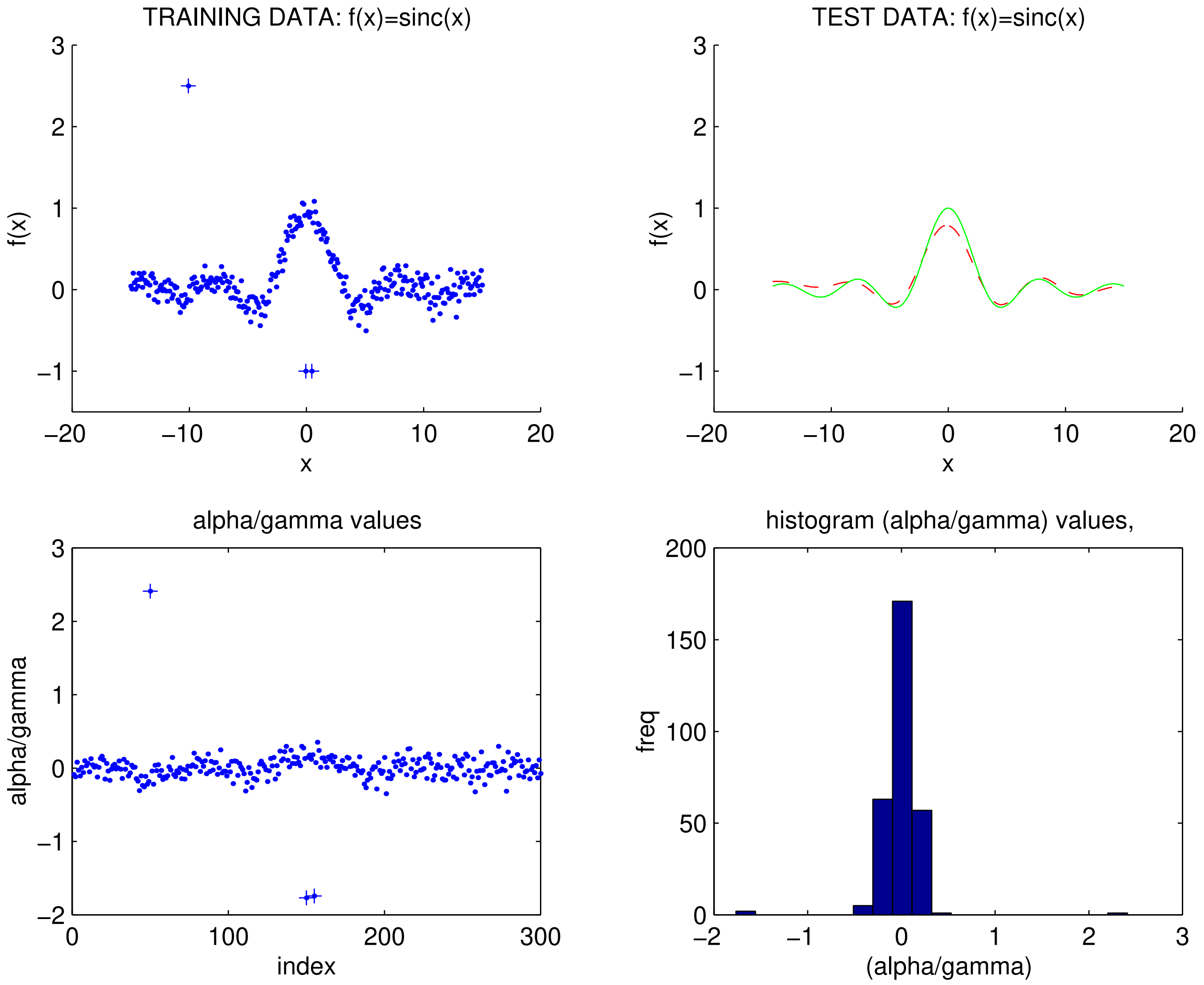
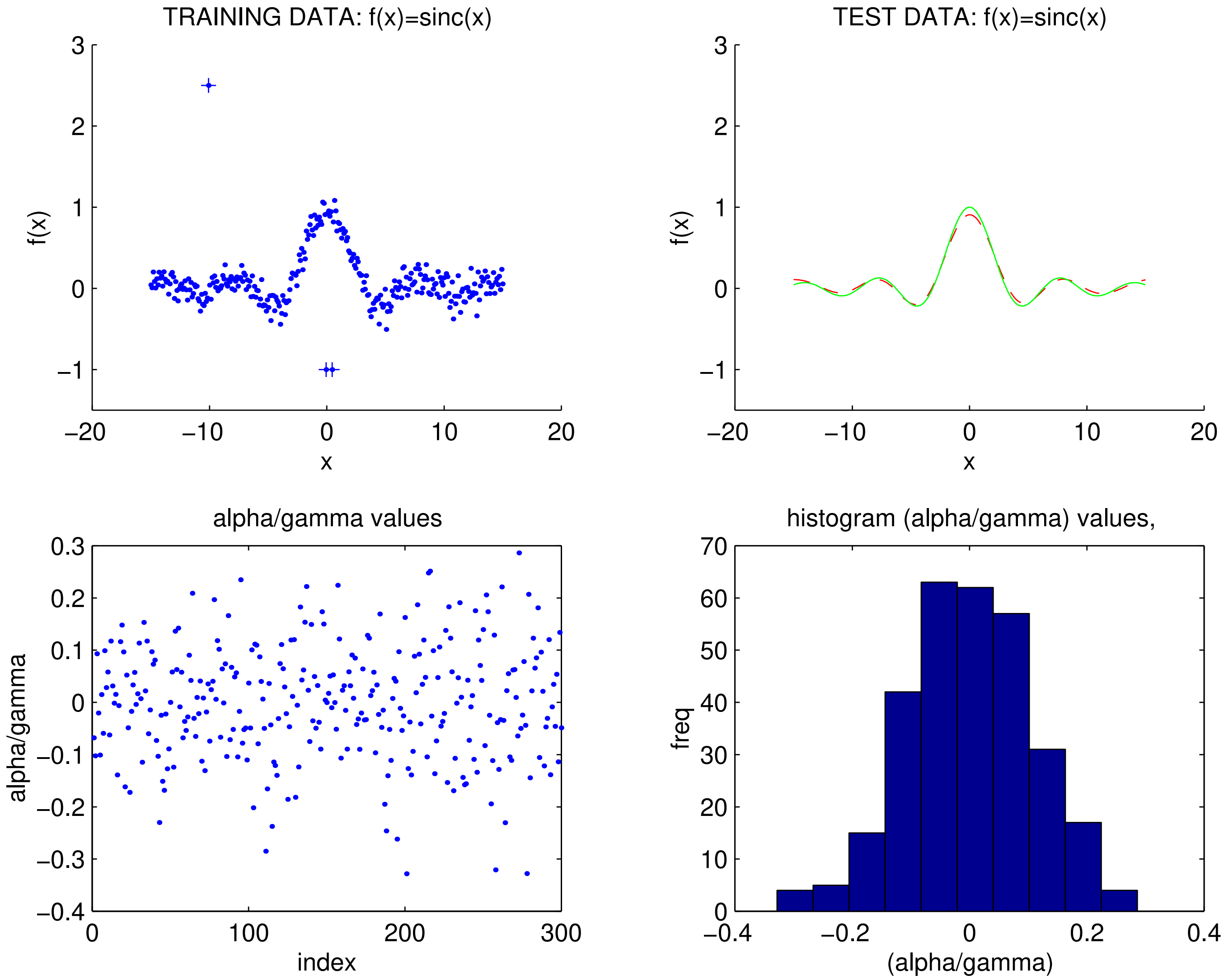
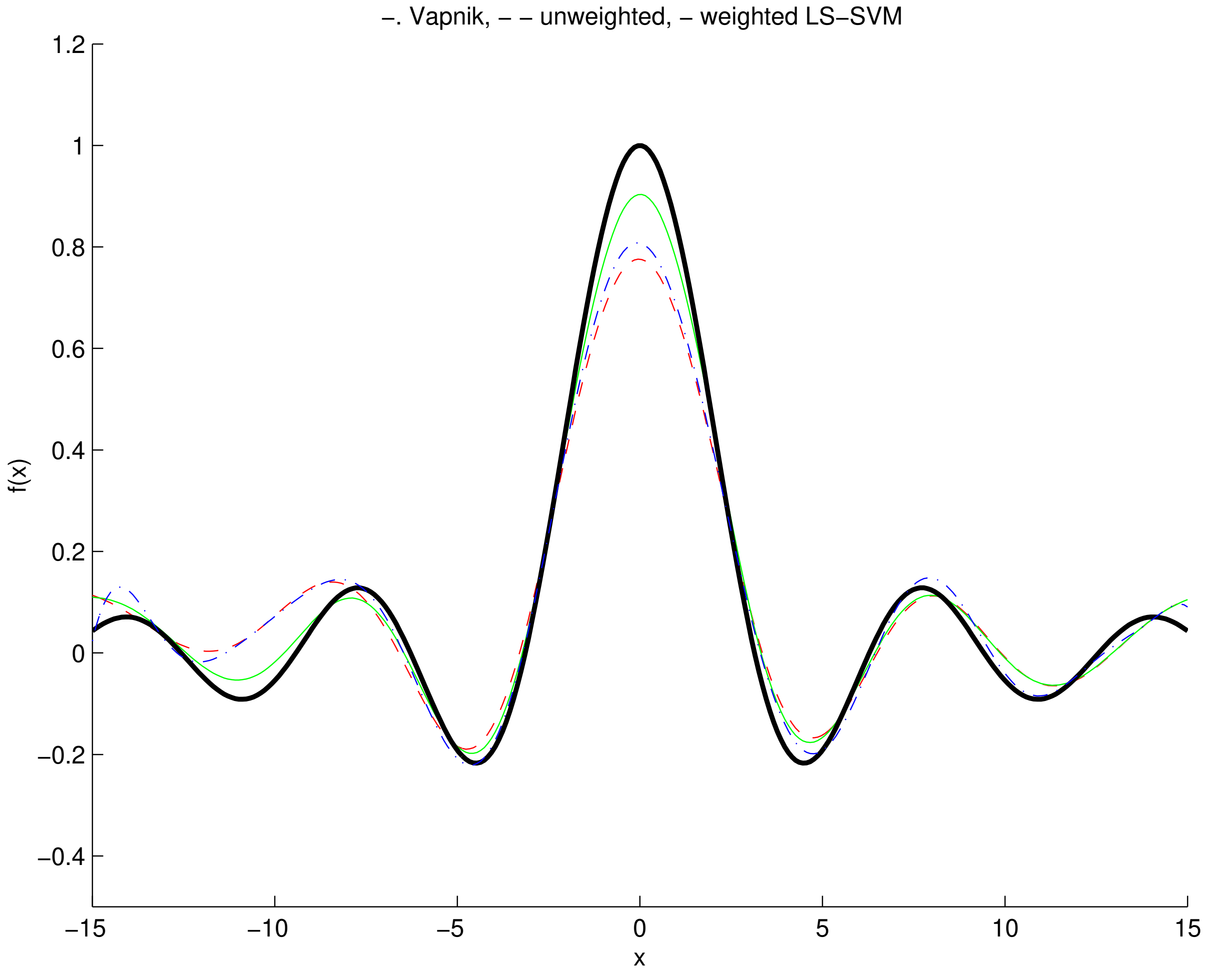

TRAINING DATA: f(x)=sinc(x)
f(x)
x
TEST DATA: f(x)=sinc(x)
f(x)
x
alpha/gamma values
alpha/gamma
index
histogram (alpha/gamma) values,
freq
(alpha/gamma)

Fig.1



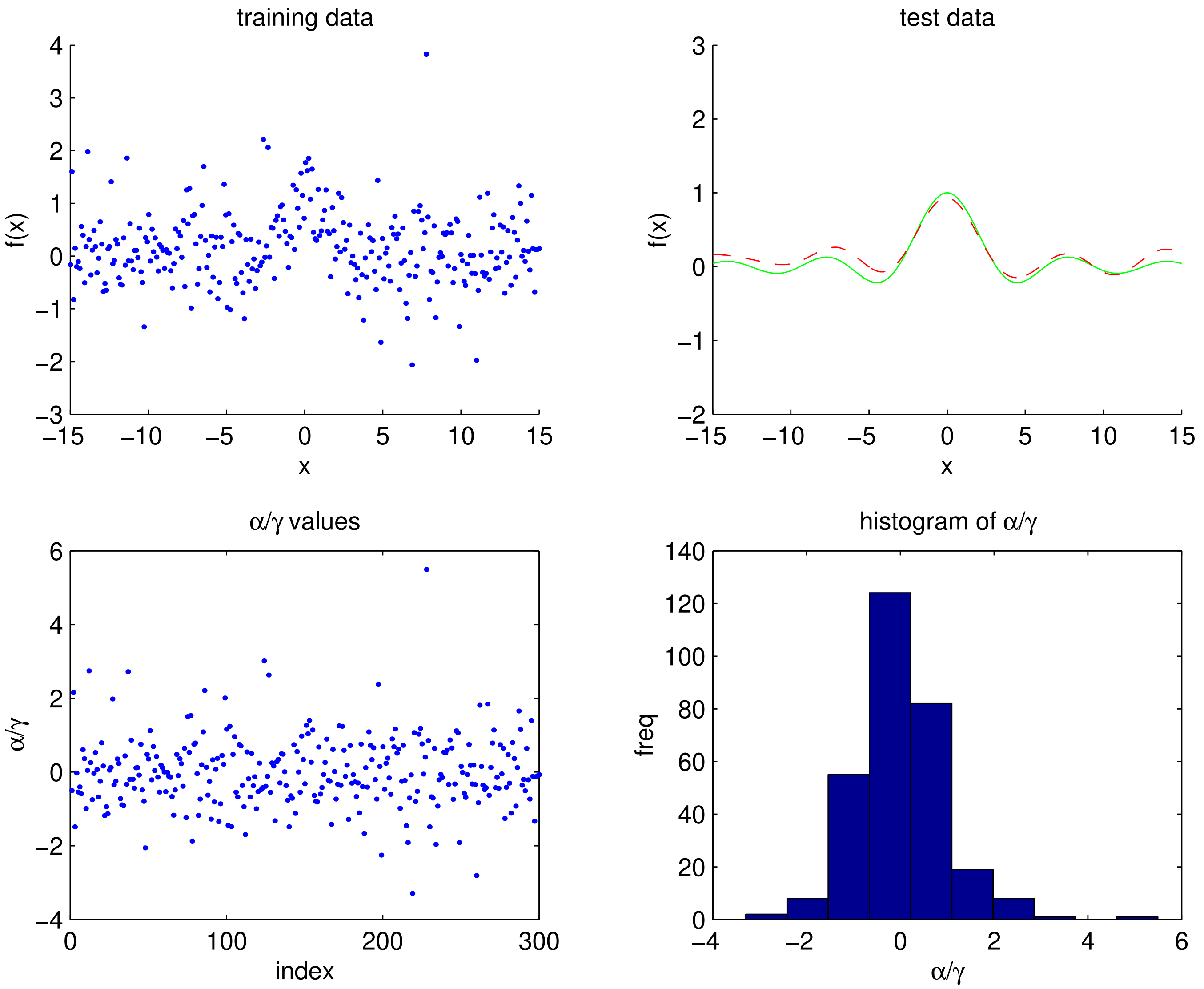
TRAINING DATA: f(x)=sinc(x)
f(x)
x
TEST DATA: f(x)=sinc(x)
f(x)
x
alpha/gamma values
alpha/gamma
index
histogram (alpha/gamma) values,
freq
(alpha/gamma)

Fig.2



−. Vapnik, − − unweighted, − weighted LS−SVM
1.2
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
f(x)
−15
−10
−5
0
5
10
15
x

Fig.3



training data
f(x)
x
test data
f(x)
x
α/γ values
α/γ
index
histogram of α/γ
freq
α/γ

Fig.4

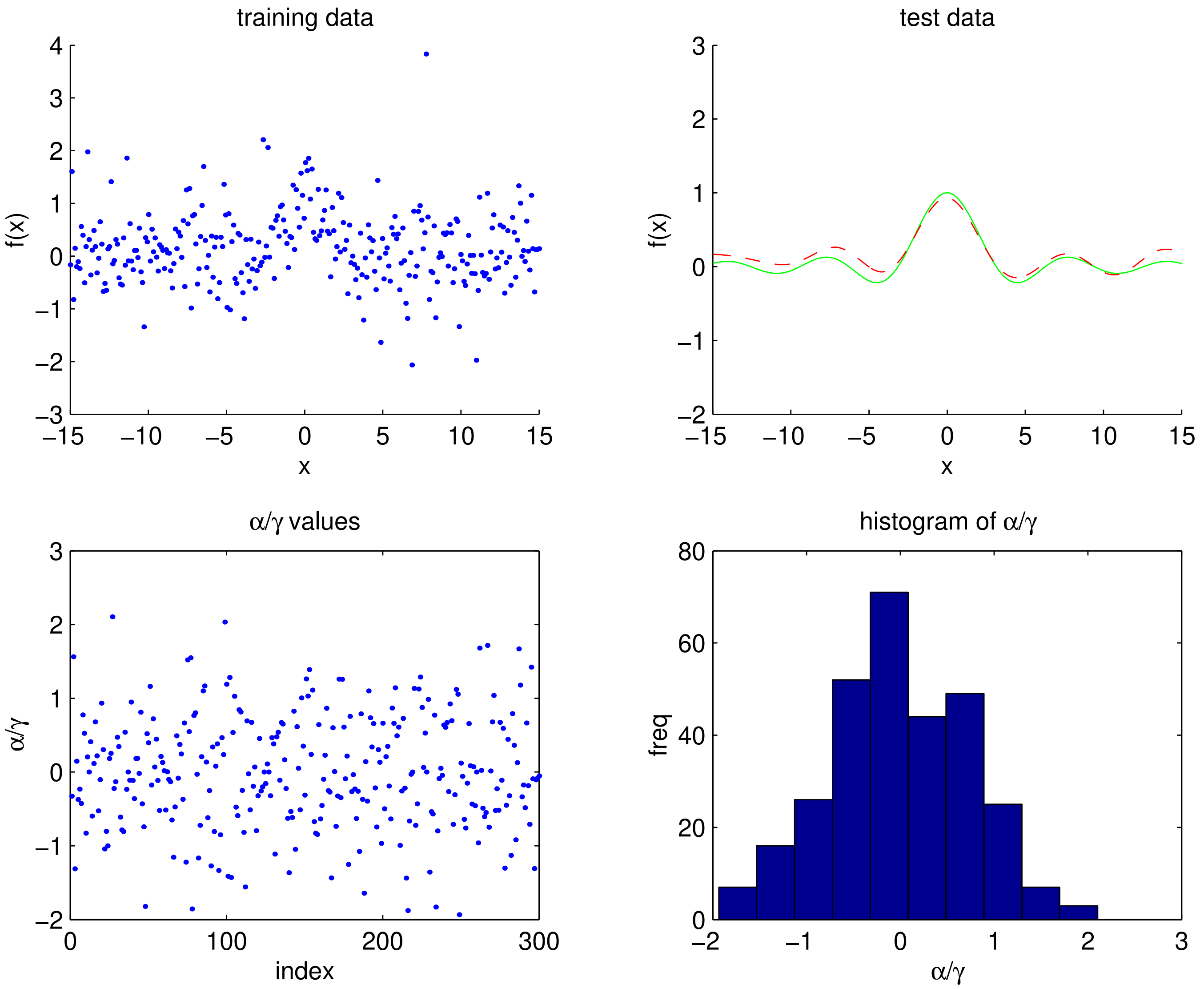
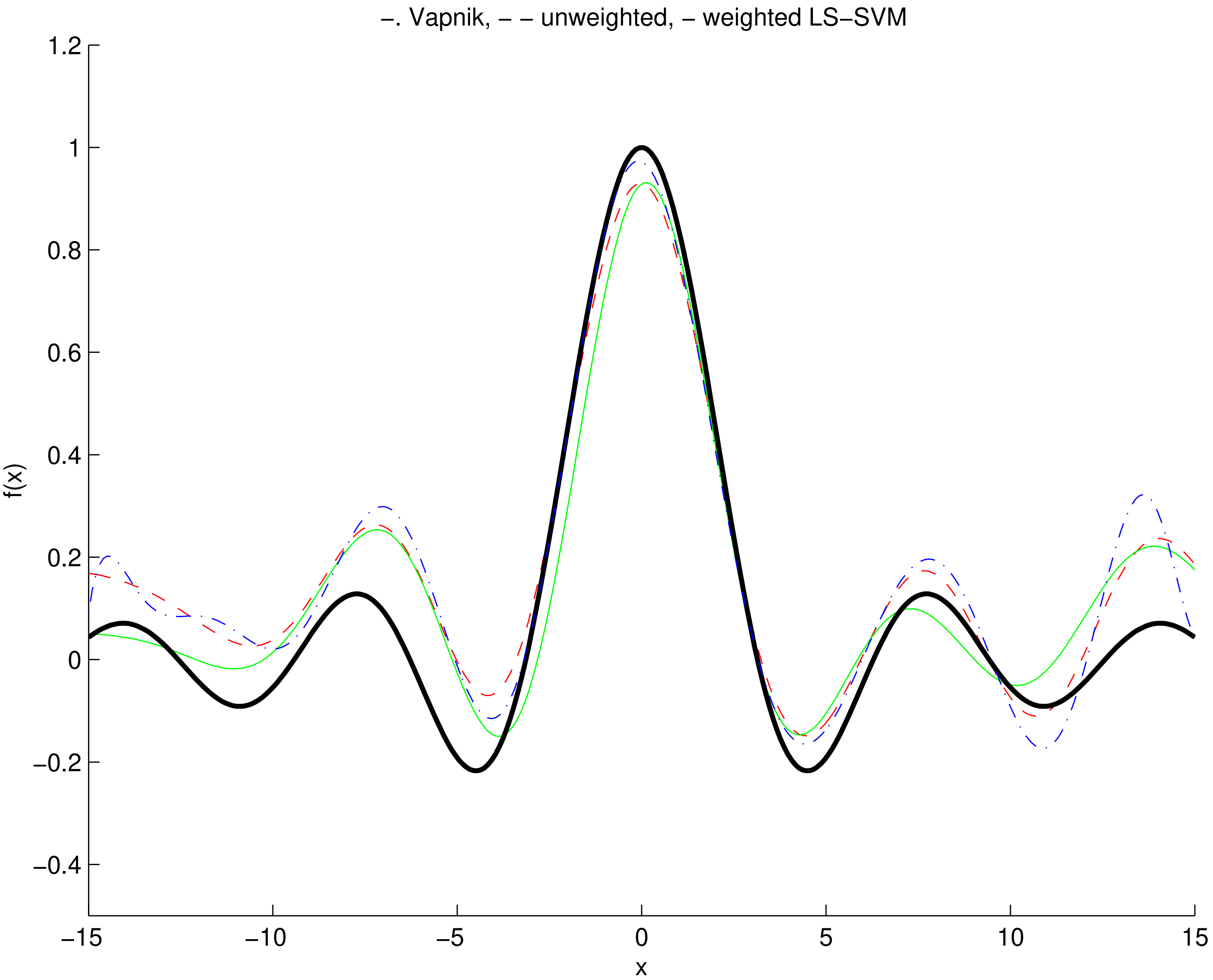
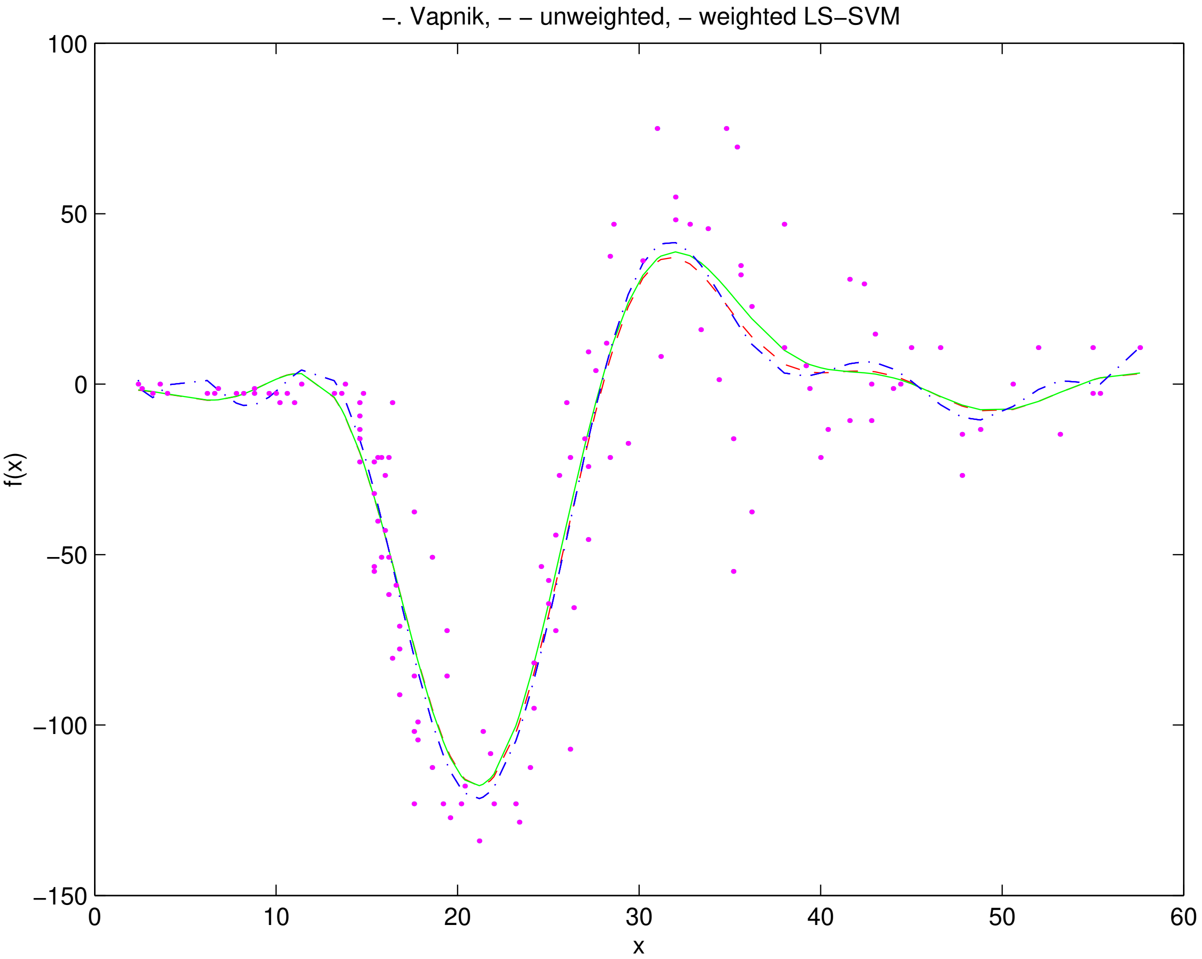

training data
f(x)
x
test data
f(x)
x
α/γ values
α/γ
index
histogram of α/γ
freq
α/γ

Fig.5



−. Vapnik, − − unweighted, − weighted LS−SVM
1.2
1
0.8
0.6
0.4
0.2
0
−0.2
−0.4
f(x)
−15
−10
−5
0
5
10
15
x

Fig.6



−. Vapnik, − − unweighted, − weighted LS−SVM
100
50
0
−50
−100
−150
f(x)
0
10
20
30
40
50
60
x

Fig.7

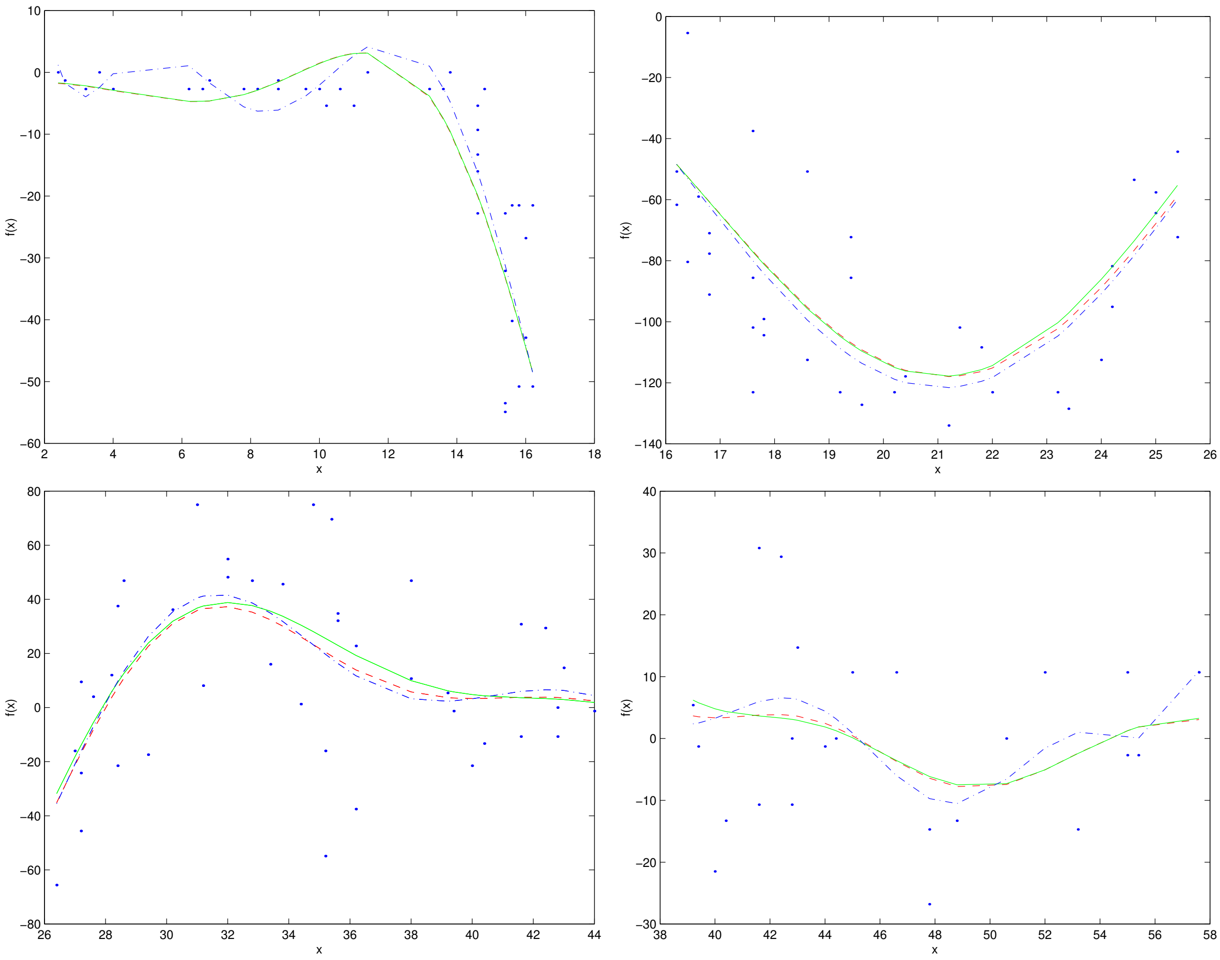


Fig.8

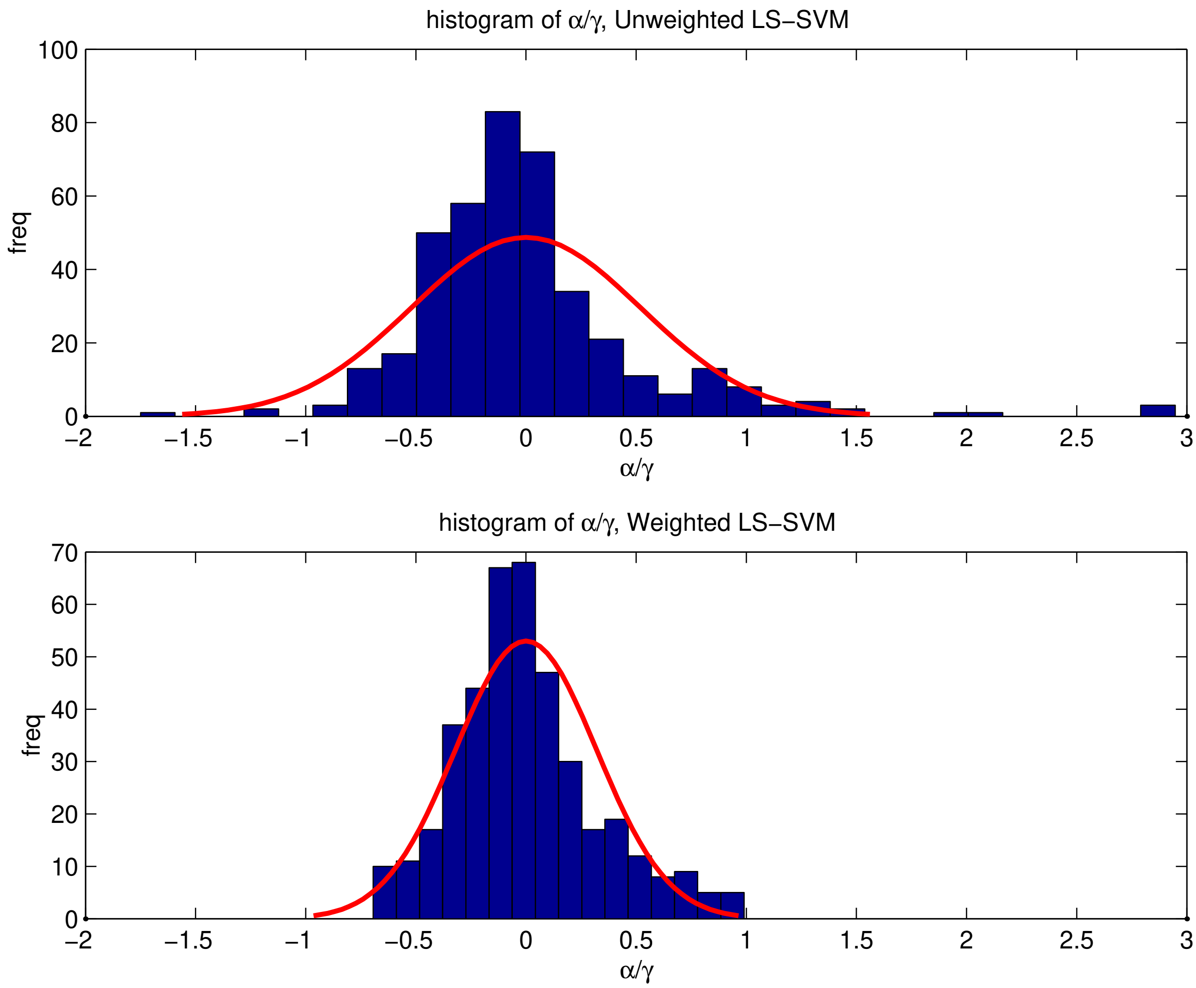

histogram of α/γ, Unweighted LS−SVM
freq
100
80
60
40
20
0
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
2.5
3
α/γ
histogram of α/γ, Weighted LS−SVM
freq
70
60
50
40
30
20
10
0
−2
−1.5
−1
−0.5
0
0.5
1
1.5
2
2.5
3
α/γ

Fig.9

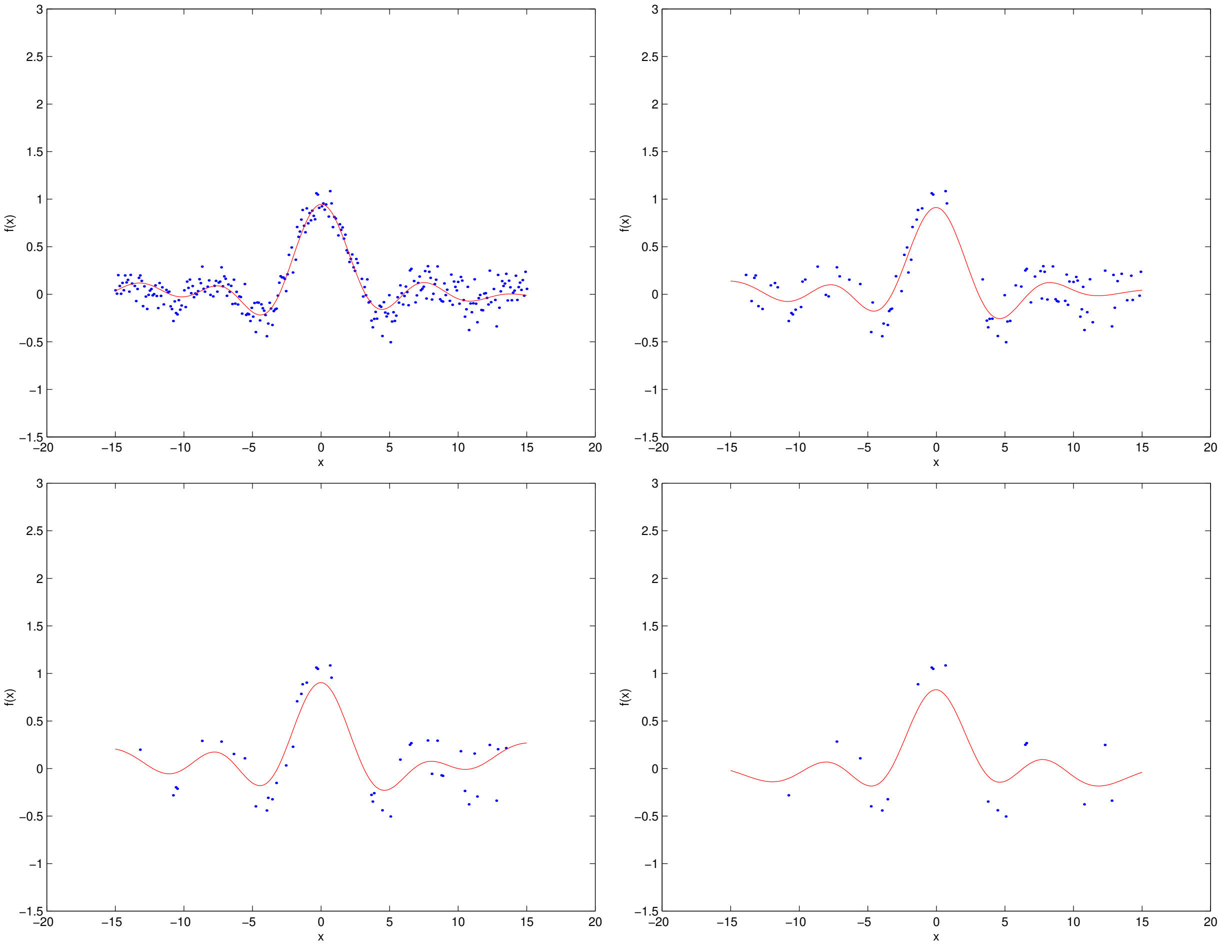


Fig.10

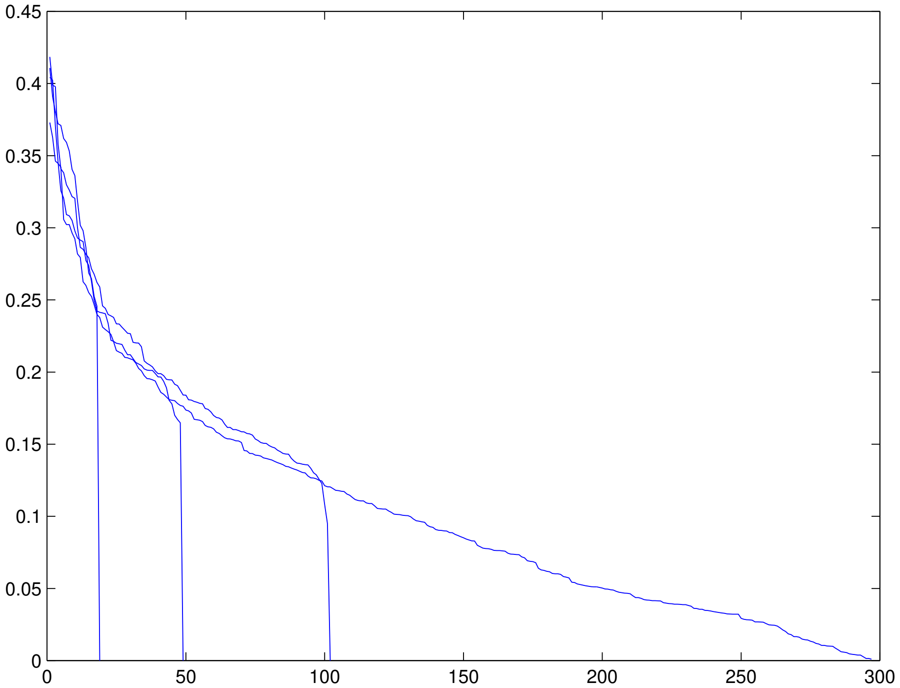


Fig.11