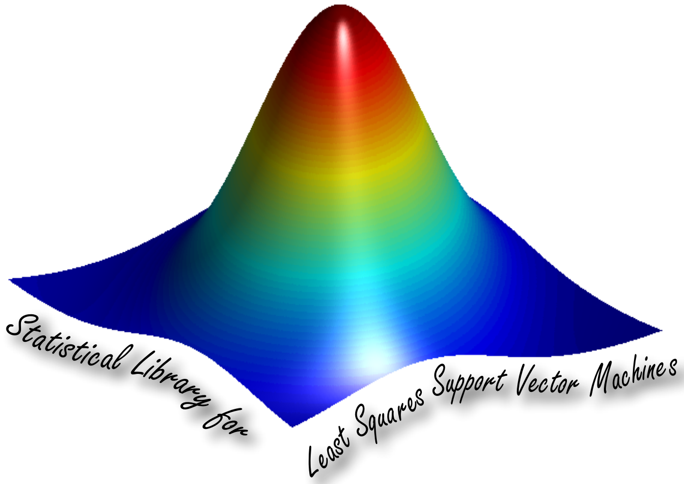
Kris De Brabanter
IOWA STATE UNIVERSITY
Department of Statistics & Computer Science
2419 Snedecor Hall, Ames, IA, 50011-1210

kbrabant@iastate.edu

Johan A.K. Suykens & Bart De Moor
KATHOLIEKE UNIVERSITEIT LEUVEN
Department of Electrical Engineering, ESAT-STADIUS
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium

{johan.suykens,bart.demoor}@esat.kuleuven.be

StatLSSVM User's Guide



October 8, 2013

Acknowledgements

Kris De Brabanter is an assistant professor at Iowa State University (Department of Statistics & Computer Science).

JS is a professor at the Katholieke Universiteit Leuven. BDM is a full professor at the Katholieke Universiteit Leuven. Research supported by Research Council KUL: GOA/10/09 MaNet , PFV/10/002 (OPTEC), several PhD/postdoc & fellow grants; Flemish Government: IOF: IOF/KP/SCORES4CHEM, FWO: PhD/postdoc grants, projects: G.0320.08 (convex MPC), G.0557.08 (Glycemia2), G.0588.09 (Brain-machine), G.0377.09 (Mechatronics MPC); G.0377.12 (Structured systems) research community (WOG: MLDM), IWT: PhD Grants, projects: Eureka-Flite+, SBO LeCoPro, SBO Climaqs, SBO POM, EUROSTARS SMART, iMinds 2012, Belgian Federal Science Policy Office: IUAP P7/ (DYSCO, Dynamical systems, control and optimization, 2012-2017) EU: ERNSI, FP7-EMBOCON (ICT-248940), FP7-SADCO (MC ITN-264735), ERC ST HIGHWIND (259 166), ERC AdG A-DATADRIVE-B, COST: Action ICO806: IntelliCIS.

Chapter 1

Overview of functions in `StatLSSVM`

Function Call	**Short explanation**	**Page**
`bimodNW`	Internal function. Estimate a regression curve based on bimodal kernels	
`bitreverse32`	Internal function. Reverse the bits of the argument	
`changelssvm`	Change a field of the object oriented representation of the LS-SVM	9
`cilssvm`	Constructs pointwise or simultaneous confidence intervals	10
`crossval`	Estimate performance of an LS-SVM with fast v-fold CV	11
`crossval2lp1`	Estimate performance of an LS-SVM with leave-$2l+1$-out CV	12
`csa`	Coupled simulated annealing with variance control	13
`cvl`	Calculate l for leave-$2l+1$-out CV	14
`gcrossval`	Estimate performance of an LS-SVM with generalized CV	15
`hall`	Estimate error variance model free using Hall's estimator	16
`huber`	Huber's loss function	17
`initlssvm`	Initiate LS-SVM object interface	18
`kernel_matrix`	Construct the positive (semi-) definite and symmetric kernel matrix	19
`kernel_matrix2`	Internal function. Fast evaluation of the kernel matrix	
`latticeseq_b2`	Internal function. Generate points from a lattice sequence in base 2 in radical inverse ordering.	
`leaveoneout`	Estimate performance of an LS-SVM with fast leave-one-out CV	20
`linf`	L_∞ norm	17
`lssvmMATLAB`	Internal function. Matlab implementation of the LS-SVM solver	
`mae`	Mean absolute error	17
`mse`	Mean squared error	17
`plotlssvm`	Plot LS-SVM results	21
`plotlssvmadd`	Plot additive LS-SVM results	22
`progress`	Internal function. Text progress bar	
`rcrossval`	Estimate performance of an LS-SVM with robust v-fold CV	23
`robustlssvm`	Robust training of LS-SVM via iterative reweighting	24
`rsimplex`	Derivative-free simplex method for robust tuning	25
`simlssvm`	Evaluate LS-SVM at arbitrary points	26
`simplex`	Derivative-free simplex method for standard LS-SVM tuning	25
`smootherlssvm`	Calculate smoother matrix for LS-SVM	27
`tbform`	Volume-of-tube formula	28
`trainlssvm`	training of LS-SVM	29
`tunelssvm`	Find suitable tuning parameters of LS-SVM	30
`weightingscheme`	Internal function. Weighting functions used by `robustlssvm`	

Chapter 2

Overview of data sets in `StatLSSVM`

All the data sets below can be found on http://www.uow.edu.au/~mwand/webspr/data.html except for the **`faithful`**, **`birth`** and **`beluga`** data sets. The latter can be found on http://research.microsoft.com/en-us/um/people/cmbishop/prml/webdatasets/faithful.txt and http://pages.stern.nyu.edu/~jsimonof/SmoothMeth/Data/Tab/ respectively.

Data set	Description and characteristics
`diabetes`	Factors affecting patterns of insulin-dependent diabetes mellitus in children. The response measurement is the logarithm of C-peptide concentration (pmol/ml) at diagnosis, and the predictor measurements are age and base deficit (a measure of acidity). The data set consist out of 43 measurements and has three columns: • `age` : age of the children • `base deficit` : measure of acidity • `Cpeptide` : response measurement
`LIDAR`	The `LIDAR` data set has 221 observations from a light detection and ranging (LIDAR) experiment. The data set contains the following columns: • `range` : distance travelled before the light is reflected back to its source • `logratio` : logarithm of the ratio of received light from two laser sources
`UStemp`	The UStemp data set has 56 observations on the temperature and location of 56 U.S. cities. This data frame contains the following columns: • `latitude` : degrees latitude (north of Equator) • `longitude` : (negative) degrees longitude (west of Greenwich) • `min.temp` : average minimum January temperature
`fossil`	The fossil data set has 106 observations on fossil shells. This data frame contains the following columns: • `age` : age in millions of years • `strontium.ratio` : ratios of strontium isotopes

`nba`	The NBA data set has 96 observations on mean points scored per minute conditional on the number of minutes played per game and height in centimeters for 96 NBA players who played the guard position during the 1992-1993 season. • `mpg` : minutes played per game • `height` : height in centimeters • `mps` : mean points scored per minute
`nba2`	The NBA2 data set has 96 observations assists per minute and points per minute for 96 NBA players who played the guard position during the 1992-1993 season. • `apm` : assists per minute • `ppm` : points per minutes
`birth`	U.S. monthly birth rate for the period from January 1940 through December 1947. This data set contains 96 observations. • `year` : year of birth • `birthrate` : birth rate
`beluga`	Nursing time (in seconds) of a newborn beluga whale calf Hudson to the time after birth, where time is measured is six-hour time periods. This data set contains 228 observations. • `period` : six-hour time periods • `nursingtime` : Nursing time (in seconds)

Chapter 3

General notation

In the full syntax description of the function calls, a star (*) indicates that the argument is optional. In the description of the arguments, a (*) denotes the default value. In this extended help of the function calls of `StatLSSVM`, a number of symbols and notations return in the explanation and the examples. These are defined as follows:

Variables	**Explanation**
`d`	Dimension of the input vectors
`m`	Dimension of the output vectors
`n`	Number of training data
`nt`	Number of test data
`X`	`N`×`d` matrix with the inputs of the training data
`Xt`	`Nt`×`d` matrix with the inputs of the test data
`Y`	`N`×`m` matrix with the outputs of the training data
`Yt`	`Nt`×`m` matrix with the outputs of the test data

This toolbox supports an object oriented interface. This has a few dedicated structures which will appear many times:

Structure	**Explanation**
`model`	Object oriented representation of the LS-SVM model

Chapter 4

Alphabetical list of function calls

4.1 `changelssvm`

Purpose

Change a field of the object oriented representation of the LS-SVM

Description and full syntax

The fields of the model structure can be altered by this function.

```
>> model = changelssvm(model,'field','value')
```

An alternative to change the field(s) is to use

```
>> model.<field> = <value>
```

The different options are given in the following table:

- General options representing the kind of model:

```
         status: Status of this model ('trained'  or 'changed' )
          alpha: Support values of the trained LS-SVM model
              b: Bias term of the trained LS-SVM model
       duration: Number of seconds the training lasts
            gam: Regularisation parameter
    kernel_type: Kernel function
      bandwidth: bandwidth of the kernel function
        weights: Weighting function for robust regression
```

- Fields used to specify the data:

```
          x_dim: Dimension of input space
          y_dim: Dimension of responses
        nb_data: Number of training data
         xtrain: Inputs of training data
         ytrain: Outputs of training data
```

See also:

`initlssvm`

4.2 `cilssvm`

Purpose

Construction of bias corrected $100(1-\alpha)\%$ *pointwise or simultaneous confidence intervals*

Description

This function calculates bias corrected $100(1-\alpha)\%$ pointwise or simultaneous confidence intervals. The bias is estimated by the principle of double smoothing with a fourth order kernel based on the Gaussian. The procedure supports homoscedastic data as well as heteroscedastic data. The construction of pointwise confidence intervals are based on the central limit theorem for linear smoothers combined with bias correction and variance estimation. The volume-of-tube formula is used for the construction of simultaneous confidence intervals. In this case, the bands are expanded to account for bias rather than recentered to obtain proper coverage.

Full syntax

```
>> ci = cilssvm(model)
>> ci = cilssvm(model, alpha)
>> ci = cilssvm(model, alpha, conftype)
>> ci = cilssvm(model, alpha, conftype,vartype)
```

Outputs

`ci`	$n \times 2$ matrix containing the lower and upper confidence intervals

Inputs

`model`	Object oriented representation of the LS-SVM model
`alpha`(*)	Significance level (by default 5%)
`conftype`(*)	Type of confidence interval 'pointwise' or 'simultaneous' (by default 'simultaneous')
`vartype`(*)	'homoscedastic' or 'heteroscedastic' (by default 'homoscedastic')

See also:

`trainlssvm, simlssvm, tbform`

References

[1] Hall P. & Marron S. (1990), On variance estimation in nonparametric regression, *Biometrika*, 77, 415-419.

[2] Sun J. & Loader C.R. (1994), Simultaneous confidence bands for linear regression and smoothing, *Annals of Statistics*, 22(3), 1328-1345.

[3] Krivobokova, T., Kneib, T. & Claeskens, G. (2010). Simultaneous confidence bands for penalized spline estimators. *Journal of the American Statistical Association*, 105(490), 852–863.

[4] De Brabanter K., De Brabanter J., Suykens J.A.K. & De Moor B. (2011), Approximate confidence and prediction intervals for least squares support vector regression. *IEEE Transactions on Neural Networks*, 22(1), 110–120.

4.3 `crossval`

Purpose

Estimate performance of an LS-SVM with fast v-fold cross-validation

Full syntax

This function is a fast implementation of v-fold CV which uses previously computed results. `crossval` can only be used in combination with `tunelssvm` (see p. 30). The command can be invoked as follows:

```
>> model = tunelssvm(model,'crossval')
```

See also:

`leaveoneout, gcrossval, rcrossval, crossval2lp1, tunelssvm`

References

[1] An S., Liu W., & Venkatesh S. (2007), Fast cross-validation algorithms for least squares support vector machine and kernel ridge regression. *Pattern Recognition*, 40(8), 2154–2162

[2] De Brabanter K., De Brabanter J., Suykens J.A.K. & De Moor, B. (2010), Optimized fixed-size kernel models for large data sets. *Computational Statistics & Data Analysis*, 54(6), 1484–1504

4.4 `crossval2lp1`

Purpose

Estimate performance of an LS-SVM with leave-$2l+1$-out cross-validation in the presence of correlated errors

Full syntax

This function is a fast implementation of leave-$2l+1$-out cross-validation which uses previously computed results. `crossval2lp1` can only be used in combination with `tunelssvm` (see p. 30). The command can be invoked as follows:

```
>> model = tunelssvm(model,'crossval2lp1')
```

See also:

`leaveoneout, crossval, gcrossval, rcrossval, tunelssvm`

References

[1] Chu, C.K. and Marron, J.S. (1991). Comparison of two bandwidth selectors with dependent errors. *Annals of Statistics*, 19(4), 1906-1918.

[2] De Brabanter K., De Brabanter J., Suykens J.A.K., De Moor B. (2011), Kernel regression in the presence of correlated errors. *Journal of Machine Learning Research*, 12, pp. 1955–1976.

4.5 `csa`

Purpose

Coupled simulated annealing finds the minimum of the functions `crossval, rcrossval, gcrosval,` `leaveoneout` *and* `crossval2lp1` *when used with* `tunelssvm`*.*

Description

The optimization process consists out of two steps: first, determine good initial starting values by means of coupled simulated annealing (CSA) and second, perform a fine-tuning derivative-free simplex search using the previous end result as starting values. In contrast to other global optimization techniques CSA is not slow and can easily escape from local minima. Since its working principle is based on coupled multiple starters it is more effective than multi-start gradient descent optimization algorithms. Another advantage of CSA is that it uses the acceptance temperature to control the variance of the acceptance probabilities with a control scheme that can be applied to an ensemble of optimizers. This leads to an improved optimization efficiency because it reduces the sensitivity of the algorithm to the initialization parameters while guiding the optimization process to quasi-optimal runs. Because of the effectiveness of the combined methods only a small number of iterations iterations are needed to acquire a suitable set of smoothing parameters (bandwidth h of the kernel and the regularization parameter γ).

References

[1] Xavier-de-Souza S., Suykens J.A.K., Vandewalle J. & Bolle D. (2010), Coupled simulated annealing. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 40(2), 320–335.

[2] Xavier-de-Souza S., *Optimisation and Robustness of Cellular Neural Networks*, PhD thesis, Faculty of Engineering, K.U.Leuven (Leuven, Belgium), Jun. 2007, 229 p.

4.6 `cvl`

Purpose

Find an estimate of l in leave-$2l+1$-out cross-validation based on bimodal kernels.

Description

Finds an estimate of l in leave-$2l+1$-out cross-validation for regression with correlated errors. The method is based on the use of bimodal kernels [2] and the fact that the quantity

$$\left|\frac{\sum_{i=1}^{n-q} \hat{e}_i \hat{e}_{i+q}}{\sum_{i=1}^{n} \hat{e}_i^2}\right| \leq \frac{\Phi^{-1}(1-\frac{\alpha}{2})}{\sqrt{n}},$$

where $\hat{e}_i$ denotes the ith residual, $q \geq 1$, Φ^{-1} denotes the quantile function of the standard normal distribution and α is the significance level [1].

Full syntax

```
>> [l,index] = cvl(X,Y)
```

Outputs

`l`	Optimal l in leave-$2l+1$-out CV
`index`(*)	Indices for leave-$2l+1$-out CV used in `crossval2lp1` (depends on l)

Inputs

`X`	Input data
`Y`	Output data

See also:

`crossval2lp1`

References

[1] Kendall M.G., Stuart A. & Ord J.K. (1983), *The Advanced Theory of Statistics, vol. 3, Design and Analysis, and Time-Series* (4th ed.), Griffin, London.

[2] De Brabanter K., De Brabanter J., Suykens J.A.K., De Moor B. (2011), Kernel regression in the presence of correlated errors. *Journal of Machine Learning Research*, 12, pp. 1955–1976.

4.7 `gcrossval`

Purpose

Estimate performance of an LS-SVM with generalized cross-validation

Full syntax

This function is a fast implementation of generalized CV which uses previously computed results. `gcrossval` can only be used in combination with `tunelssvm` (see p. 30). The command can be invoked as follows:

```
>> model = tunelssvm(model,'gcrossval')
```

See also:

`leaveoneout, crossval, rcrossval, crossval2lp1, tunelssvm`

References

[1] Craven P. & Wahba G. (1979), Smoothing noisy data with spline functions, *Numerische Mathematik*, 31(4), 377-403.

[2] Golub G.M., Heath M. & Wahba G. (1979), Generalized cross-validation as a method for choosing a good ridge parameter, *Technometrics*, 21(2), 215-223.

[3] De Brabanter K., *Least Squares Support Vector Regression with Applications to Large-Scale Data: a Statistical Approach*, PhD thesis, Faculty of Engineering, K.U.Leuven (Leuven, Belgium), Apr. 2011, 246 p.

4.8 `hall`

Purpose

Calculate the error variance model free using Hall's estimator

Description

Calculate the error variance in 1d model free using Hall's estimator which is $\sqrt{n}$-consistent. Consider the data $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$. First, the input data is sorted, i.e. $X_1 \leq \ldots \leq X_n$, and the output data is sorted accordingly. Second, the estimated error variance $\hat{\sigma}^2$ is calculated as

$$\hat{\sigma}^2 = \frac{1}{n-2} \sum_{i=1}^{n-2} (0.809 Y_i - 0.5 Y_{i+1} - 0.309 Y_{i+2})^2.$$

Full syntax

```
>> var = hall(X,Y)
```

Outputs

`var`	Estimated error variance

Inputs

`X`	Input data (one dimensional)
`Y`	Output data

References

[1] Hall P. & Marron S. (1990), On variance estimation in nonparametric regression, *Biometrika*, 77, 415–419.

[2] Hall P., Kay J.W. & Titterington D.M. (1990), Asymptotically optimal difference-based estimation of variance in nonparametric regression, *Biometrika*, 77(3), 521–528.

4.9 huber, linf, mae, mse

Purpose

Loss functions

Description

A variety of cost measures can be defined

- `huber` $$C_{\text{huber,c}}(e) = \begin{cases} \frac{1}{2}e^2, & |e| \leq c; \\ 2c|e| - c^2, & \text{otherwise.} \end{cases}$$
- `mae`: L_1 $C_{L_1}(e) = \frac{\sum_{i=1}^{n} |e_i|}{n}$
- `linf`: L_∞ $C_{L_\infty}(e) = \sup_i |e_i|$
- `mse`: L_2 $C_{L_2}(e) = \frac{\sum_{i=1}^{n} e_i^2}{n}$

The parameter c in Huber's loss function can be user specified or automatically determined (default). If no user specific value is supplied, the parameter is calculated by $1.345\text{MAD}(e)$.

Full syntax

- `>> C = huber(e)`
  `>> C = huber(e,c)`

  Outputs

`C`	Estimated cost

  Inputs

`e`	Input vector, for example residuals (one dimensional)
`c`(*)	User specified parameter c

- `>> C = mse(e)`

  Outputs

`C`	Estimated cost of the residuals

  Calls

`mse`	`mae`, `linf` or `mse`

  Inputs

`e`	Residual vector

See also:

`crossval, rcrossval, gcrossval, crossval2lp1, leaveoneout`

References

[1] Huber P.J. (1964). Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1), 73-101.

4.10 `initlssvm`

Purpose

Initialize an LS-SVM model

Full syntax

```
>> model = initlssvm(X, Y, gam, h, kernel)
>> model = initlssvm(X, Y, gam, h, kernel, bwopt)
```

Outputs

`model`	Object oriented representation of the LS-SVM model

Inputs

`X`	`n`×`d` matrix with the inputs of the training data
`Y`	`n`×`1` vector with the outputs of the training data
`gam`	Regularization parameter. Initialize as `[]`
`h`	Kernel bandwidth. Initialize as `[]`)
`kernel`(*)	Kernel type (by default `'gauss_kernel'`). The following kernels are supported: `'gauss_kernel'`, `'RBF_kernel'` `'gauss4_kernel'`, `'gaussadd_kernel'` and `'lin_kernel'`.
`bwopt`(*)	Single bandwidth (`'single'`) or bandwidth per dimension (`'multiple'`) for additive models. Only valid with `'gaussadd_kernel'`. By default `'single'`

See also:

`changelssvm`

4.11 `kernel_matrix`

Purpose

Construct the positive (semi-) definite and symmetric kernel matrix

Basic Syntax

```
>> Omega = kernel_matrix(X, kernel, h)
```

Description

This matrix should be positive definite if the kernel function satisfies Mercer's condition. Construct the kernel values for all test data points in the rows of `Xt`, relative to the points of `X`.

```
>> Omega_Xt = kernel_matrix(X, kernel, h, Xt)
```

The following kernels are supported

- Gaussian kernel (`gauss_kernel`):

  $$K(X_i, X_j) = (2\pi)^{-d/2} \exp\left(-\frac{\|X_i - X_j\|_2^2}{2h^2}\right).$$

- RBF kernel (`RBF_kernel`): (numerically more stable than `gauss_kernel` when d is large)

  $$K(X_i, X_j) = \exp\left(-\frac{\|X_i - X_j\|_2^2}{2h^2}\right).$$

- Fourth order kernel based on the Gaussian (`gauss4_kernel`):

  $$K(X_i, X_j) = \frac{1}{2}(2\pi)^{-d/2}\left(3 - \frac{\|X_i - X_j\|_2^2}{h^2}\right)\exp\left(-\frac{\|X_i - X_j\|_2^2}{2h^2}\right).$$

- Gaussian additive kernel (`gaussadd_kernel`):

  $$\Omega(X_i, X_j) = \sum_{k=1}^{d} K(X_i^{(k)}, X_j^{(k)}),$$

  where $K(\cdot,\cdot)$ is the one dimensional Gaussian kernel and $X_i^{(k)}$ denotes the kth component of the d-dimensional column vector X_i.

- Linear kernel (`lin_kernel`):

  $$K(X_i, X_j) = X_i^T X_j.$$

Full syntax

```
>> Omega = kernel_matrix(X, kernel, h)
>> Omega = kernel_matrix(X, kernel, h, Xt)
```

Outputs

`Omega`	`n`×`n` (`n`×`nt`) kernel matrix

Inputs

`X`	`n`×`d` matrix with the inputs of the training data
`kernel`	Kernel type (by default `'gauss_kernel'`)
`h`	Kernel bandwidth (for linear kernel, use `[]`)
`Xt`(*)	`nt`×`d` matrix with the inputs of the test data

4.12 `leaveoneout`

Purpose

Estimate performance of an LS-SVM with fast leave-one-out cross-validation

Full syntax

This function is a fast implementation of leave-one-out CV which uses previously computed results. **`leaveoneout`** can only be used in combination with **`tunelssvm`** (see p. 30). The command can be invoked as follows:

```
>> model = tunelssvm(model,'leaveoneout')
```

See also:

`crossval, gcrossval, rcrossval, crossval2lp1, tunelssvm`

References

[1] Ying Z. & Keong K.C. (2004), Fast leave-one-out evaluation and improvement on inference for LS-SVMs, in *Proc of the 17th International Conference on Pattern Recognition (ICPR)*, Volume 3, pp. 494 - 497.

4.13 `plotlssvm`

Purpose

Plot the LS-SVM results in the environment of the training data

Basic syntax

```
>> model = plotlssvm(model)
```

Description

The first argument specifies the LS-SVM. If the model does not have status `'trained'`, the training algorithm is first called. One can specify the precision of the plot by specifying the `grain` of the grid. By default this value is 100. The dimensions (`seldims`) of the input data to display can be selected as an optional argument in case of higher dimensional inputs (> 2). A grid will be taken over this dimension, while the other inputs remain constant.

Full syntax

```
>> model = plotlssvm(model)
>> model = plotlssvm(model, plottype)
>> model = plotlssvm(model, plottype, grain)
>> model = plotlssvm(model, plottype, grain, seldims)
```

Outputs

`model`(*)	Trained object oriented representation of the LS-SVM model

Inputs

`model`	Object oriented representation of the LS-SVM model
`plottype`(*)	Specifies type of plot for 2d problems (`surface` or `contour` plot) (by default `surf`)
`grain`(*)	The grain of the grid evaluated to compose the surface (by default `100`)
`seldims`(*)	The principal inputs one wants to span a grid (by default `[1 2]`)

See also:

`trainlssvm, simlssvm, plotlssvmadd`.

4.14 `plotlssvmadd`

Purpose

Plot the additive LS-SVM results in the environment of the training data

Basic syntax

```
>> model = plotlssvm(model)
```

Description

The first argument specifies the LS-SVM. If the model does not have status `'trained'`, the training algorithm is first called. The output of the function is a plot of each fitted function per dimension. This function works only if the number of dimensions is larger or equal than two. Bands representing twice the pointwise standard error of the estimated curve are also visualized together with the partial residuals i.e. the fitted values for each function plus the overall residuals from the additive model.

Full syntax

```
>> model = plotlssvmadd(model)
>> model = plotlssvmadd(model, axislabels)
```

Outputs

`model`(*)	Trained object oriented representation of the LS-SVM model

Inputs

`model`	Object oriented representation of the LS-SVM model
`axislabels`(*)	Cell of axis names, e.g. {'age','pressure'}. By default each x-axis is labeled as $X_1, \ldots X_d$ and the y-axis $m_1(X_1), \ldots, m_d(X_d)$

See also:

`trainlssvm, simlssvm, plotlssvm`.

4.15 `rcrossval`

Purpose

Estimate performance of an LS-SVM with fast robust cross-validation

Full syntax

This function is a fast implementation of robust CV which uses previously computed results. `rcrossval` can only be used in combination with `tunelssvm` (see p. 30). It is advised always to use an L_1 loss (`mae`) for this type of CV. The command can be invoked as follows:

```
>> model = tunelssvm(model,'rcrossval',{10,'mae'})
```

See also:

`leaveoneout, gcrossval, crossval, crossval2lp1, tunelssvm`

References

[1] De Brabanter K., Pelckmans K., De Brabanter J., Debruyne M., Suykens J.A.K., Hubert M. & De Moor B. (2009), Robustness of kernel based regression: a comparison of iterative weighting schemes, in *Proc. of the 19th International Conference on Artificial Neural Networks (ICANN)*, pp. 100-110

[2] De Brabanter K., Karsmakers P., De Brabanter J., Pelckmans K., Suykens J.A.K. & De Moor B. (2010), On robustness in kernel based regression, *NIPS 2010 Workshop Robust Statistical Learning.*

[3] De Brabanter K. (2011), *Least Squares Support Vector Regression with Applications to Large-Scale Data: a Statistical Approach*, PhD thesis, Faculty of Engineering, K.U.Leuven (Leuven, Belgium), Apr. 2011, 246 p.

4.16 `robustlssvm`

Purpose

Robust training in the case of non-Gaussian noise or outliers

Basic syntax

```
>>  model  = robustlssvm(model)
```

Robustness towards outliers can be achieved by reducing the influence of support values corresponding to large errors. One should first use the function `tunelssvm` with `rcrossval` based on a robust loss function (`mae` or `huber`) so all the necessary parameters are optimally tuned before calling this routine.

Full syntax

```
>> model = robustlssvm(model)
```

Outputs

`model`	Robustly trained object oriented representation of the LS-SVM model

Inputs

`model`	Object oriented representation of the LS-SVM model

See also:

`trainlssvm, tunelssvm, rcrossval`

References

[1] De Brabanter K., Pelckmans K., De Brabanter J., Debruyne M., Suykens J.A.K., Hubert M. & De Moor B. (2009), Robustness of kernel based regression: a comparison of iterative weighting schemes, in *Proc. of the 19th International Conference on Artificial Neural Networks (ICANN)*, pp. 100-110.

[2] M. Debruyne, A. Christmann, M. Hubert & J.A.K. Suykens (2010), Robustness of reweighted least squares kernel based regression, *Journal of Multivariate Analysis*, 101(2), 447–463.

[3] De Brabanter K., Karsmakers P., De Brabanter J., Pelckmans K., Suykens J.A.K. & De Moor B. (2010), On robustness in kernel based regression, *NIPS 2010 Workshop Robust Statistical Learning.*

[4] De Brabanter K. (2011), *Least Squares Support Vector Regression with Applications to Large-Scale Data: a Statistical Approach*, PhD thesis, Faculty of Engineering, K.U.Leuven (Leuven, Belgium), Apr. 2011, 246 p.

4.17 `rsimplex, simplex`

Purpose

Direct search method that does not use numerical or analytic gradients.

Description

The optimization process consists out of two steps: first, determine good initial starting values by means of coupled simulated annealing (CSA) and second, perform a fine-tuning derivative-free simplex search using the previous end result as starting values. This is a direct search method that does not use numerical or analytic gradients. If l is the length of a vector x, a simplex in l-dimensional space is characterized by the $l+1$ distinct vectors that are its vertices. In two-space, a simplex is a triangle; in three-space, it is a pyramid. At each step of the search, a new point in or near the current simplex is generated. The function value at the new point is compared with the function's values at the vertices of the simplex and, usually, one of the vertices is replaced by the new point, giving a new simplex. This step is repeated until the diameter of the simplex is less than the specified tolerance. `rsimplex` and `simplex` are the same functions except `rsimplex` is specifically used for robust regression in collaboration with `rcrossval`.

Because of the effectiveness of the combined methods only a small number of iterations are needed to acquire a suitable set of smoothing parameters (bandwidth h of the kernel and the regularization parameter γ).

References

[1] Nelder J.A. & Mead R. (1965), A simplex method for function minimization, *Computer Journal*, 7, 308–313.

[2] Lagaria J.C., Reeds J.A., Wright M.H. & Wright P.E. (1998), Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions, *SIAM Journal of Optimization*, 9(1), 112–147.

4.18 `simlssvm`

Purpose

Evaluate the LS-SVM at given points

Basic syntax

```
>> Yt = simlssvm(model, Xt)
```

Description

`model` is the object oriented representation of the LS-SVM model and the matrix `Xt` represents the points one wants to predict.

Full syntax

```
>> Yt = simlssvm(model, Xt)
```

Outputs

`Yt`	Vector with predicted output of test data

Inputs

`model`	Object oriented representation of the LS-SVM model
`Xt`	`nt`×`d` matrix with the inputs of the test data

See also:

`trainlssvm, initlssvm, plotlssvm, plotlssvmadd`

4.19 `smootherlssvm`

Purpose

Calculate smoother matrix for LS-SVM

Basic syntax

```
>> S = smootherlssvm(model)
```

Description

LS-SVM is a linear smoother because it can be represented as $\hat{Y} = SY$ where S is the smoother matrix. This matrix can be used to calculate the error variance and confidence intervals.

Full syntax

```
>> S = smootherlssvm(model)
>> S = smootherlssvm(model, Xt)
```

Outputs

`S`	Smoother matrix

Inputs

`model`	Object oriented representation of the LS-SVM model
`Xt`(*)	`nt`×`d` matrix with the inputs of the test data. If supplied, `S` is the smoother matrix for test data

See also:

`trainlssvm, cilssvm, tbform`

References

[1] De Brabanter K., De Brabanter J., Suykens J.A.K. & De Moor B. (2011), Approximate confidence and prediction intervals for least squares support vector regression, *IEEE Transactions on Neural Networks*, 22(1), 110–120.

[2] De Brabanter K. (2011), *Least Squares Support Vector Regression with Applications to Large-Scale Data: a Statistical Approach*, PhD thesis, Faculty of Engineering, K.U.Leuven (Leuven, Belgium), Apr. 2011, 246 p.

4.20 `tbform`

Purpose

Calculate width of the bands for simultaneous confidence intervals using the volume-of-tube formula.

Basic syntax

```
>> m = tbform(model)
```

Description

Determines the width of the bands, given a significance level α, for simultaneous/uniform confidence intervals based on the volume-of-tube formula.

Full syntax

```
>> m = tbform(model)
>> m = tbform(model,alpha)
```

Outputs

`m`	Width of the bands

Inputs

`model`	Object oriented representation of the LS-SVM model
`alpha`(*)	significance level. By default 0.05

See also:

`smootherlssvm, cilssvm`

References

[1] Rice S.O. (1939), The distribution of the maxima of a random curve, *American Journal of Mathematics* 61, 409-416.

[2] Sun J. & Loader C.R. (1994), Simultaneous confidence bands for linear regression and smoothing, *Annals of Statistics*, 22(3), 1328-1345.

[3] De Brabanter K. (2011), *Least Squares Support Vector Regression with Applications to Large-Scale Data: a Statistical Approach*, PhD thesis, Faculty of Engineering, K.U.Leuven (Leuven, Belgium), Apr. 2011, 246 p.

4.21 `trainlssvm`

Purpose

Train the support values and the bias term of an LS-SVM

Basic syntax

```
>> model      = trainlssvm(model)
```

Description

The training is done by

```
>> model = trainlssvm(model)
```

The status of the model checks whether a retraining is needed. The implementation is based on the backslash operator in `MATLAB`. This operator solves the system of linear equations needed to obtain the Lagrange multipliers and bias term of the LS-SVM model.

Full syntax

```
>> model = trainlssvm(model)
```

Outputs

`model`(*)	Trained object oriented representation of the LS-SVM model

Inputs

`model`	Object oriented representation of the LS-SVM model

See also:

`simlssvm, initlssvm, changelssvm, plotlssvm, plotlssvmadd`

4.22 `tunelssvm`

Purpose

Tune the tuning parameters of the model with respect to the given performance measure

Basic syntax

```
>> model = tunelssvm(model, costfun)
```

where `model` is the object oriented interface of the LS-SVM and `costfun` represents the model selection criteria e.g. `crossval`, `gcrossval`, `leaveoneout`, `crossval2lp1` or `rcrossval`. The `model` object is created by the command `initlssvm`.

```
model = initlssvm(X,Y,[],[],kernel_type);
```

Description

The complete tuning process goes as follows: First, for every kernel, first Coupled Simulated Annealing (CSA) determines suitable starting points for every method. The search limits of the CSA method are set to $[\exp(-10), \exp(10)]$. Second, these starting points are then given to a simplex method. CSA has already proven to be more effective than multi-start gradient descent optimization. Another advantage of CSA is that it uses the acceptance temperature to control the variance of the acceptance probabilities with a control scheme. This leads to an improved optimization efficiency because it reduces the sensitivity of the algorithm to the initialization parameters while guiding the optimization process to quasi-optimal runs. By default, CSA uses five multiple starters. When the model is tuned, the value of the cross-validation function is added the to model structure as `model.CVcost`.

Full syntax

```
>> model = tunelssvm(model)
>> model = tunelssvm(model, costfun)
>> model = tunelssvm(model, costfun, costargs)
>> model = tunelssvm(model, costfun, costargs, wfun)
```

Outputs

`model`	Tuned object oriented representation of the LS-SVM model

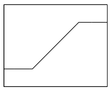
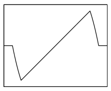
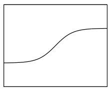
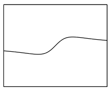
Inputs

`model`	Object oriented representation of the LS-SVM model
`costfun`(*)	Model selection criterion. By default `'crossval'`
`costargs`(*)	Cell with extra cost function arguments. By default `{10,'mse'}` where the first argument denotes the number of folds to be used in the `'crossval'` routine and the second argument specifies its loss function
`wfun`(*)	Weight function for robust regression (by default `'wmyriad'`). The weight function can only be used with `'rcrossval'`. Supported weight functions: `'whuber'`, `'whampel'`, `'wlogistic'`, `'wmyriad'`

The four different weights functions are visualized in Table 4.1

When no options are specified for the model selection criteria, the software automatically takes an L_2 (L_1) loss with 10-fold cross-validation if `crossval` (or `rcrossval`) is taken respectively. The following list summarizes the possibilities with the `tunelssvm` command.

Table 4.1: Definitions for the Huber ($\beta \in \mathbb{R}^+$), Hampel, Logistic and Myriad ($\delta \in \mathbb{R}_0^+$) weight functions $V(\cdot)$. The corresponding loss $L(\cdot)$ and score function $\psi(\cdot)$ are also given.

	Huber	Hampel	Logistic	Myriad
$V(r)$	$\begin{cases} 1, & \text{if } \lvert r\rvert < \beta; \\ \frac{\beta}{\lvert r\rvert}, & \text{if } \lvert r\rvert \geq \beta. \end{cases}$	$\begin{cases} 1, & \text{if } \lvert r\rvert < b_1; \\ \frac{b_2-\lvert r\rvert}{b_2-b_1}, & \text{if } b_1 \leq \lvert r\rvert \leq b_2; \\ 0, & \text{if } \lvert r\rvert > b_2. \end{cases}$	$\frac{\tanh(r)}{r}$	$\frac{\delta^2}{\delta^2+r^2}$
$\psi(r)$				
$L(r)$	$\begin{cases} r^2, & \text{if } \lvert r\rvert < \beta; \\ \beta\lvert r\rvert - \frac{1}{2}\beta^2, & \text{if } \lvert r\rvert \geq \beta. \end{cases}$	$\begin{cases} r^2, & \text{if } \lvert r\rvert < b_1; \\ \frac{b_2 r^2-\lvert r^3\rvert}{b_2-b_1}, & \text{if } b_1 \leq \lvert r\rvert \leq b_2; \\ 0, & \text{if } \lvert r\rvert > b_2. \end{cases}$	$r\tanh(r)$	$\log(\delta^2+r^2)$

- Standard tuning with 10-fold cross-validation with an L_2 and L_1 loss

  ```
  >> model = tunelssvm(model)
  >> model = tunelssvm(model,’crossval’)
  >> model = tunelssvm(model,’crossval’,{10,’mse’})
  >> model = tunelssvm(model,’crossval’,{10,’mae’})
  ```

- Standard tuning with leave-one-out cross-validation with an L_2 and L_1 loss

  ```
  >> model = tunelssvm(model,’leaveoneout’)
  >> model = tunelssvm(model,’leaveoneout’,{’mse’})
  >> model = tunelssvm(model,’leaveoneout’,{’mae’})
  ```

- Standard tuning with generalized cross-validation with an L_2 and L_1 loss

  ```
  >> model = tunelssvm(model,’gcrossval’)
  >> model = tunelssvm(model,’gcrossval’,{’mse’})
  >> model = tunelssvm(model,’gcrossval’,{’mae’})
  ```

- Robust tuning with different weight functions with robust v-fold cross-validation with an L_1 and Huber loss. If no weight function is specifies myriad weights are taken. It is always advisable to use a robust loss function for robust cross-validation (`mae` or `huber`) in order to obtain a fully robust procedure.

  ```
  >> model = tunelssvm(model,’rcrossval’,{10,’mae’})
  >> model = tunelssvm(model,’rcrossval’,{10,’mae’},’wmyriad’)
  >> model = tunelssvm(model,’rcrossval’,{10,’huber’},’whampel’)
  >> model = tunelssvm(model,’rcrossval’,{10,’mae’},’whuber’)
  >> model = tunelssvm(model,’rcrossval’,{10,’huber’},’wlogistic’)
  ```

- Tuning when errors are correlated with an L_2 and L_1 loss

  ```
  >> model = tunelssvm(model,’crossval2lp1’)
  >> model = tunelssvm(model,’crossval2lp1’,{’mse’})
  >> model = tunelssvm(model,’crossval2lp1’,{’mae’})
  ```

See also:

`simlssvm, initlssvm, changelssvm, crossval, gcrossval, rcrossval, leaveoneout, crossval2lp1`