

Iterative Single Data Algorithm (ISDA) Manual

ISDA (Copyright (C) 2004-2007 Te-Ming Huang and Vojislav Kecman)

is an efficient software for solving **large-scale** machine learning (data mining) problems by using so-called kernel machines a.k.a. support vector machines. ISDA solves two classic *supervised* learning from data (statistics) problems – classification (a.k.a. pattern recognition) and multivariable regression (a.k.a. curve, surface and hyper-surface fitting, a.k.a. approximation). ISDA algorithm solves exactly quadratic programming (QP) based learning problem developed for kernel machines in an iterative way. It belongs to the class of *working set* solvers (same as, for example, the Sequential Minimal Optimization (SMO) algorithm). Theoretical foundations of the ISDA algorithm are presented and can be found in following publications:

1. Kecman V., Vogt M., Huang T.-M., *On the Equality of Kernel AdaTron and Sequential Minimal Optimization in Classification and Regression Tasks and Alike Algorithms for Kernel Machines*, Proc. of the 11th European Symposium on Artificial Neural Networks, ESANN 2003, pp. 215 – 222, Bruges, Belgium, 2003
2. Huang T.-M., Kecman V., *Bias Term b in SVMs Again*, Proc. of the 12th European Symposium on Artificial Neural Networks, ESANN 2004, pp. 441-448, Bruges, Belgium, 2004
3. Kecman V., T.-M. Huang, M. Vogt, *Chapter ‘Iterative Single Data Algorithm for Training Kernel Machines from Huge Data Sets: Theory and Performance’*, in a Springer-Verlag book, ‘Support Vector Machines: Theory and Applications’, Ed. L. Wang, Series: Studies in Fuzziness and Soft Computing, Vol. 177, pp. 255-274, 2005
4. Huang T.-M., V. Kecman, I. Kopriva, *Kernel Based Algorithms for Mining Huge Data Sets, Supervised, Semi-supervised, and Unsupervised Learning*, Springer-Verlag, Berlin, Heidelberg, 2006, see www.learning-from-data.com

References 1) and 2) can be downloaded from the site www.support-vector.ws while 3) and 4) can be found on the Springerlink site, or in the mentioned books. The site www.learning-from-data.com contains more information on ISDA including the download of ISDA algorithms for both classification and regression written in Matlab. Please note that ISDA is free for academic purposes. For commercial purposes please contact authors.

In a supervised learning one uses a set of input-output training data pairs to design a decision function in a classification or an approximating function in regression. Thus, a data set $D = \{[\mathbf{x}(i), y(i)] \in \mathcal{R}^m \times \mathcal{R}, i = 1, \dots, n\}$ consists of **n pairs** $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_n, y_n)$, where the inputs \mathbf{x} are the m -dimensional vectors $\mathbf{x} \in \mathcal{R}^m$ and, system responses are either integer values $y \in [1, 2, 3, \dots]$ in classification, or they are continuous values $y \in \mathcal{R}$ in regression problems. (Note that even in the case of two classes i.e., for dichotomization tasks, **we don’t use labeling +1 and -1**).

For designing a good model one needs both a training data set and a test data set. Usually, the available data set D is split in two sets by using 80% or 90% or 95% randomly chosen data pairs from the data set D as a training data set and the rest of 20% or 10% or 5% of the data is used for the test. (Percentages are problem dependant.). Thus, test data are data which the learning machine has not seen during the training phase. After the training, the test data is used to calculate a performance of a model expressed by a suitable norm or cost function (usually the number of misclassified data in classification and, sum of error squares or sum of absolute values or similar

cost function in regression is used). The best model is the one having minimal error norm on the test data. Note that during the running of ISDA you will be asked to provide both data sets.

Format of data

ISDA accepts data in several formats, notably the ones in csv format and the ones either in a sparse format or in a dense one as given below. A standard csv formatted data can be created in Excel, Matlab, or in other data creating software. Note that only the dense data can use csv format.

For both classification and regression data must be organized as follows, the first column must be the desired (target) vector \mathbf{y} and the rest of columns are the columns of the input vector \mathbf{x} . Thus, if you solve the problem having 1,000 training data and the input vectors \mathbf{x}_i (vectors of features) are 3-dimensional your training data set must be organized as below

$$\begin{bmatrix} y_1 & x_{11} & x_{12} & x_{13} \\ y_2 & x_{21} & x_{22} & x_{23} \\ \vdots & \vdots & \vdots & \vdots \\ y_{1,000} & x_{1,000\ 1} & x_{1,000\ 2} & x_{1,000\ 3} \end{bmatrix}$$

In the case of a regression y_i is the measured (observed, sampled, recorded) value of the output when the input was $[x_{i1} \ x_{i2} \ x_{i3}]$. Similarly, for a classification y_i is a label (class belonging) of the input $[x_{i1} \ x_{i2} \ x_{i3}]$.

The csv (Comma Separated Values) is a standard Excel format. Note that you can create data in a csv format directly in Matlab as follows: Assume your training data are organized in a matrix as shown above and the matrix has the name `datatrain`. The command

```
dlmwrite('C:\ISDA\data_for_training.csv',datatrain)
```

will create a file `data_for_training.csv` within the ISDA folder which can be used by ISDA

Labeling in Classification

Note that **ISDA can solve multi-class problems** and not only the two-class (a.k.a. dichotomization) one. This defines the labels used in ISDA software. **The label y_i must be i if the input (vector of features) $\mathbf{x}_i = [x_{i1} \ x_{i2} \ x_{i3}]$ belongs to class i .** Thus, for example if the first data belongs to class 2, second one to class 4, third one to class 3, and the fourth one to the class 5, the first four entries to vector \mathbf{y} are $[2 \ 4 \ 3 \ 5]^T$. Note that even in the case of two classes you can use neither labels $[1, 0]$ nor $[-1, 1]$. Use labeling 1 for class 1, and 2 for class 2 instead.

Sparse and Dense Input Data Format for both Classification and Regression

In addition to a csv format, ISDA can take data in both sparse and dense format as the input. For first time solving a given problem with ISDA, you need to convert your data set into these two formats or into the csv format. Here we introduce the concepts of sparse and dense data format. The only difference will be that in the sparse format zeros (0) are not given as an input making in this way data files much smaller if many input components are zeros.

Assume you have 7 data originating from a three-class problem and the input is 4 dimensional as given below by their numerical values

$$\begin{bmatrix} 1 & 0 & 1.1 & 0.3 & -1.1 \\ 2 & -2 & 0 & 1.1 & 0.7 \\ 2 & 1.1 & -3 & 0 & 1.1 \\ 2 & 0 & 0 & 0 & 2 \\ 3 & 5 & -0.5 & 1 & 2.3 \\ 2 & 2 & 0 & -4.1 & 0 \\ 2 & 0 & 1.1 & 0 & 3.7 \end{bmatrix}$$

Data in **DENSE** format are to be given as follows,

1	1:0	2:1.1	3:0.3	4:-1.1
2	1:-2	2:0	3:1.1	4:0.7
2	1:1.1	2:-3	3:0	4:1.1
2	1:0	2:0	3:0	4:2
3	1:5	2:-0.5	3:1	4:2.3
2	1:2	2:0	3:-4.1	4:0
2	1:0	2:1.1	3:0	4:3.7

and the data in **SPARSE** format are to be given as,

1	2:1.1	3:0.3	4:-1.1	
2	1:-2	3:1.1	4:0.7	
2	1:1.1	2:-3	4:1.1	
2	4:2			
3	1:5	2:-0.5	3:1	4:2.3
2	1:2	3:-4.1		
2	2:1.1	4:3.7		

Having the training data set and the test data one ready, one can use ISDA software for designing good kernel machines model. The GUI version of ISDA is the software that enables model selection, meaning choosing the best parameters of the model (e.g., order of polynomial, width of Gaussian kernel, penalty parameter C , size of the ε -tube in regression, etc). This is achieved by cross-validation procedure, namely by calculation of the error norm on the test data sets and by picking up the parameters that ensure the lowest error norm.

ISDA with GUI is a user friendly software but in order to make its use as smooth as possible the friendly user guidelines are presented below.

ISDA GUI USER'S GUIDE

There are few phases of using ISDA:

1. Save your isda.exe file in an appropriate folder.
2. Training phase: Choose the data you want to model which can be in the same folder as the isda.exe or in a different one. The training data file shall be opened in the first ISDA window and the test one in the second ISDA window.
3. Saving the results of the learning - after the learning (training, optimization) phase, the resulting files will be saved in the subfolder result, which will be in the same folder where the isda.exe was. The results for each parameters set will be stored under the names SVCRBF*i*.dat, SVRRBF*i*.dat, SVCPOLY*i*.dat, and SVRPOLY*i*.dat where *i* will be number going from 0 to the number of models (e.g., in classification, for $C = [1\ 10\ 100]$, and Order of Polynomial = $[1\ 2\ 3\ 4\ 5\ 6]$, $3*6 = 18$ models will be stored under the names from SVCPOLY0.dat to SVCPOLY17.dat). Obviously C and R stand for classification and regression, while RBF and POLY describe whether Gaussian kernels or Polynomial ones have been used. In addition to these files, two more files will always be stored - SVC_opt_model.dat and simulation_log.dat files. As its name says the SVC_opt_model.dat contains the model that produces the smallest test error. This file should be the model used in the so-called Prediction, or Application phase. This is the last stage in using ISDA software, and it is described below.
4. Prediction or Application phase – as its names suggests is the use of the best selected model on the new data set.

Note that there is one intermediate step between the phases 3 and 4 which works for classification problems only (because there are only two design parameters, C and either σ of Gaussians or the polynomial's order). After training, there is an option to save the current simulation results, by clicking the "Save Result" button on ISDA window. Give the name to the file, e.g. results, which will be saved in the same folder where your isda.exe routine is. By using the attached Matlab routine plot_result.m you can plot the model selection results as contour and surface plot in Matlab (by running plot_result('results')), and see in this way influence of the penalty parameter C and either width parameter σ (for Gaussian kernel) or order of polynomial (for polynomial kernel) on error cost function.

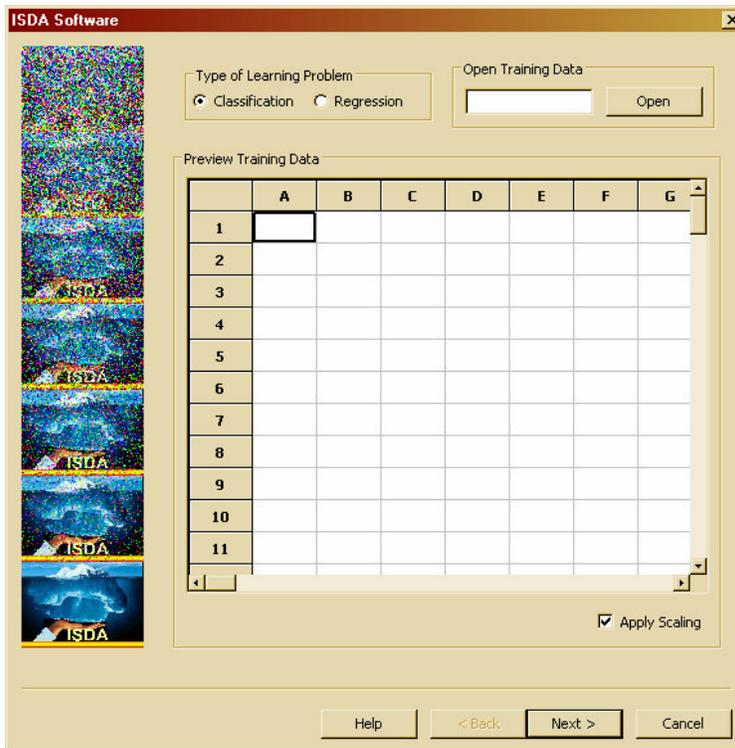
TRAINING PHASE

CLASSIFICATION

In the first ISDA window (shown next) two selections shall be done:

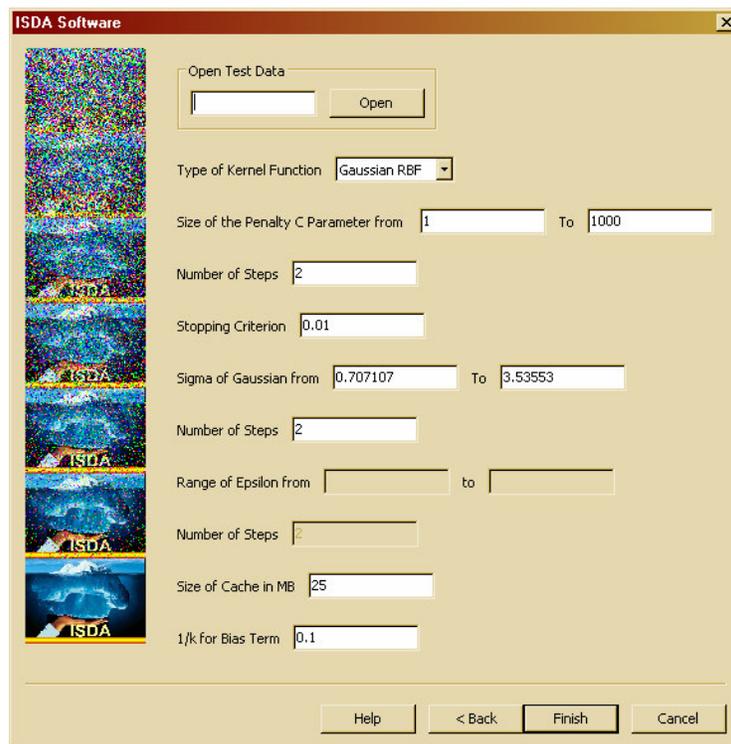
- a) classification task is selected by default,
- b) open the training data file

(Note that appearance of the windows e.g., color setting, depends upon your machine settings. Maple color scheme is shown here except the last window on page 8 which is in a classic windows color scheme).



After opening it only first 100 training data pairs will be shown. Scaling the inputs between -1 and +1 will be done unless the Apply Scaling box is unticked. We highly recommend doing the scaling. Note that you see true inputs values in the table.

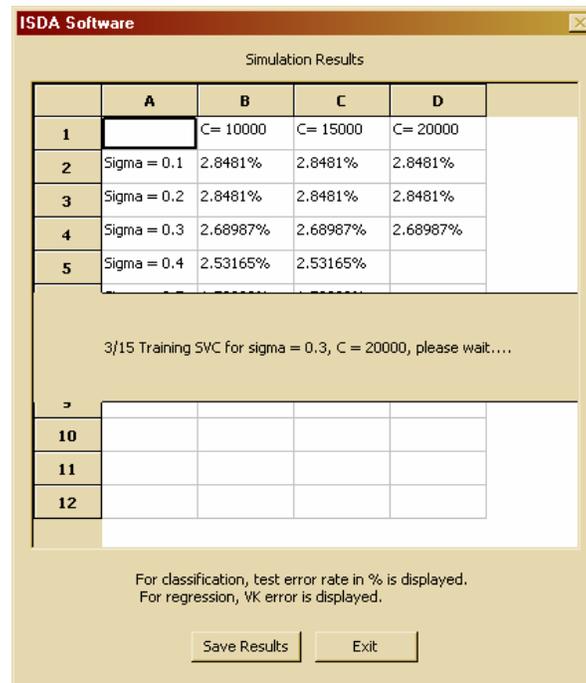
Clicking Next opens the second ISDA window where all the rest of selections shall be done as given below:



- a) Open Test Data. (The data set which will be not seen by machine during the training, and which will be used for calculation of error norm i.e., cost function, or performance),
- b) Choose what type of kernel you want to use. Choices are – Gaussian RBF kernel, Linear kernel and the Polynomial one.
- c) Select the penalty parameter C between the minimal and maximal value. Number of steps defines how many C parameters you want to use during the model optimization. Picking up Number of steps = 2 will run the calculations for minimal and maximal C value, while choosing Number of steps = 3 leads to computing for minimal C value, (minimal + maximal)/2 C value and for the maximal C, and so on. Note that you should always input the lower bound of the C value on the left hand side text box and the upper bound on the right hand side one.
- d) Stopping Criterion τ will stop updating the dual variables α_i when the differences between the desired values y_i and model outputs f_i for support vectors (SVs) become smaller than τ . Choose either 0.01 or 0.001. Higher precision leads to huge increase in computing time without any significant improvement of the model. First few simulations should be tried by $\tau = 0.01$, or even bigger, say 0.025 or 0.05 and later, after getting the feelings about the appropriate values for C and σ , or for a polynomial order, one can use $\tau = 0.001$.
- e) If Gaussian kernel is selected you will be offered two values to start with minimal one being equal to ‘average’ distance between the training data points calculated by $\sigma_0 = \frac{2}{n-1} \sqrt{m}$ (where n stands for the number of data and m is a dimensionality of the input vector) and the maximal one equals to $5\sigma_0$. You may select any other value for the two sigmas. If you had chosen polynomial kernel you will be asked to select a minimal and a maximal order of polynomial.
- f) Number of steps defines how many *sigmas* or *orders of polynomials* you want to use during the model optimization. Note that if you have picked 4 C values and 5 *sigmas* you are aiming at calculating 20 different kernel machines models from which the best one will be selected based on the merit (minimal error norm i.e., cost function).
- g) Size of cache in MB should be chosen in order to speed up the calculations (which may take a lot of time for huge data sets and/or highly overlapped classes and/or wrong choice of design parameters, or for some other reason). The most demanding situation may be if all the data are selected as the SVs. For example, if you have 5,000 training data pairs and all are chosen as SVs the memory needed for storing the Hessian matrix involved in calculations is 200 MB. On the other side if you have 500,000 training data pairs and only 25 have been chosen as SVs requires ‘only’ 100 MB for a storage. Thus, right choice of the cache size depends upon the guess what will be the final size of the model defined primarily by the number of data and number of SVs. If the cache size is too small the problem will be solved but it will take longer CPU time.
- h) $1/k$ for Bias Term - defines whether we want to have the model without bias term (then $1/k$ for Bias Term = 0) or we want to have a bias term ($1/k$ for Bias Term = 0.05,

0.1 or some other value). Note that choosing bigger values leads to worsening the condition number of the Hessian matrix involved, and thus to the longer computations. As the starter begin with $1/k$ for Bias Term = 0. Including a bias term leads to the decrease in the number of SVs, which usually results in faster training. However, the right choice of the bias is problem dependant.

Once the training phase starts *the third window (as shown below)* will be opened showing the results in a table and the progress of the training phase. In the case of classification the test error rate presented is calculated as follows $\frac{\# \text{ misclassified data}}{n_{test}}100$ i.e., it is given in %.



After the training phase, you can save the results by clicking on Save Results button and giving the name of the result file (say you gave the name results), or finish without saving the results. The single purpose of saving the results is to show the simulation results given in the table of the third window graphically within the Matlab by calling the program `plot_result('results')`. Note that this works for classification tasks only. (Sure, both the `plot_result.m` and `results` must be in the same folder which should be made a working folder in Matlab). If you are not pleased with results obtained and shown in the table and you want to do model design by other parameters selection just click Exit, and run `isda.exe` again with new parameters.

REGRESSION

ISDA software for regression operates in its biggest part same as for classification. There is only a single difference in a parameters definition procedure in respect to classification stemming from the fact that the required precision of the regression model has to be defined in advance by prescribing the size of the so-called 'ε-tube'.

First, select the Regression and open the training data on the first ISDA window.

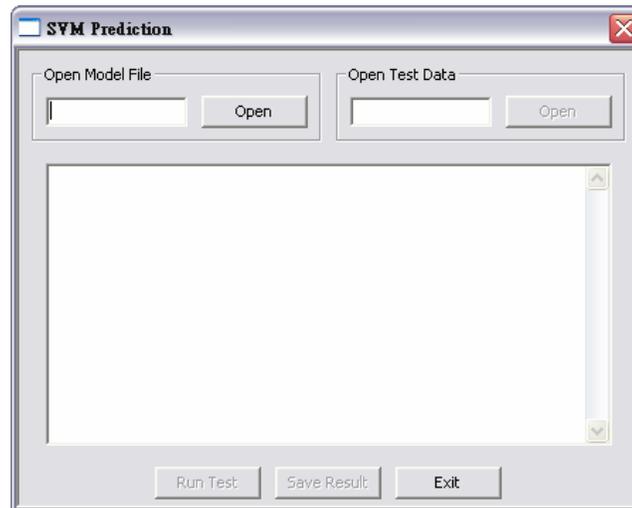
On the second ISDA windows there is only one difference in respect to the classification problem, and this is a selection of the size of the so-called Vapnik's insensitivity parameter ε which defines the ' ε -tube' within which all the training data must lie (except the so called free SVs which will be placed on the ' ε -tube' and the so-called bounded SVs that are allowed to lie outside the ' ε -tube'). Isda.exe will offer you two default values (minimal and maximal ε) which you can readily change to suit your design requirements.

Note also that, unlike in classification, you will not be able to see the results of regression in graphics form by using the program plot_result.m. However, feel free to adapt plot_result.m for plotting regression results for the best ' ε -tube', or for all the ' ε -tubes'. (If you do so send us your code please.)

Root-Mean-Squared Error (RMS) presented in the 3rd window as given above, is the error cost

function for the test data and it is calculated as $RMS = \sqrt{\frac{\sum_{i=1}^n (y_{it} - y_{ia})^2}{n}}$, where $(y_{it} - y_{ia})$ is the error or difference between the test target value y_{it} and the SVMs' approximation y_{ia} for the test input \mathbf{x}_i and n stands for the number of test data used.

PREDICTION i.e., APPLICATION PHASE



The main purpose of the predict.exe program is to apply the models developed in the training phase to the new, previously unseen, data. (Sure, you can run your best model on either training or test data just to check the performance of ISDA GUI software, and this will repeat the run on the opened data set by the best model chosen).

You will generally like to use your best model on new data sets. There are two ways how this can be done. First and the simplest, or the user the most friendly, way is to run predict.exe and open the desired data sets. Before doing any of prediction the new data set should be prepared as follows below

Preparation of the New (Test, Application) Data Set for predict.exe

The data format can be either in the csv or in the sparse and dense format described in the previous section. The first column is again equal to the desired value of the output y_i . In the case where the output y_i is unknown, i.e. the class belonging or the true output of data i is unknown, and it should be predicted by the SVMs model, the value of y_i in the test data file should be set to 0 for both classification and regression. For example, the first four points in the input matrix below are unknown where the rest are known.

$$\begin{bmatrix} 0 & 0 & 1.1 & 0.3 & -1.1 \\ 0 & -2 & 0 & 1.1 & 0.7 \\ 0 & 1.1 & -3 & 0 & 1.1 \\ 0 & 0 & 0 & 0 & 2 \\ 3 & 5 & -0.5 & 1 & 2.3 \\ 2 & 2 & 0 & -4.1 & 0 \\ 2 & 0 & 1.1 & 0 & 3.7 \end{bmatrix}$$

Note that in this case, the error rate produced by the program is no longer accurate, since the true outputs of the first four points are unknown by the program. A click on the save result button will save the outputs predicted by the SVMs model into a file.

For regression, three different measurements of errors are computed and saved in SVR_opt_model.dat - the RMS error, Mean Absolute Error (MAE) and Vojislav Kecman's error (VK error). RMS is defined above, and the other two are defined below

$$MAE = \frac{\sum_{i=1}^n |y_{it} - y_{ia}|}{n}, \quad VK \text{ error} = \frac{RMS}{e^{-|\bar{y}_t|} \sigma_{y_t} + |\bar{y}_t|} = \frac{\sqrt{\frac{\sum_{i=1}^n (y_{it} - y_{ia})^2}{n}}}{e^{-|\bar{y}_t|} \sigma_{y_t} + |\bar{y}_t|}$$

where σ_{y_t} stands for the variance of the test target vector \mathbf{y}_t and $|\bar{y}_t|$ is its absolute mean value. VK error is related to the 'level of Gaussian noise' in data. Given enough training data, spoiled by zero means Gaussian noise and providing a good SVM's model output (approximant) \mathbf{y}_a of the noiseless target vector \mathbf{y} , the value of VK error will reproduce the level of noise very good. Thus, VK error = 0.25 hints that the level of noise is 25%. Note that the same can't be said for RMS and MAE, because their values are heavily influenced by the variance σ_{y_t} and mean \bar{y}_t of the target vector \mathbf{y}_t .

In the true application case when none of the outputs is known the first column will be all zeros. After using the predict.exe you will be asked to save the result file which will contain class labels and output values for classification and regression respectively. Note that the values of error functionals at the bottom of the result file does not have any meaning in pure application case.

Often, however, you may want to have the model at hand and use it in a more flexible way (say within the Matlab, Mathematica, Mathcad, Maple, C, or some other programming language). This can readily be done by taking all the information from the resulting files SVC_opt_model.dat i.e.,

SVR_opt_model.dat stored in the result folder for a classification and regression respectively. All the basic information about the model obtained can be retrieved from these two files as follows below.

CLASSIFICATION

The file SVC_opt_model.dat contains the following information for a case of 2-dimensional input and 2 classes (dichotomization task):

```
svm_type svc
kernel_type RBF
data_type Dense
sigma 4.000000
C 10.000000
dim 2
scale_factor 0.027397 1.465753 0.400000 1.000000
nr_class 2
bias -0.051012
nSV 4
10.000000 1:-0.397260 2:1.000000
4.570504 1:-0.726027 2:-1.000000
-8.951533 1:-0.589041 2:1.000000
-10.000000 1:-0.342466 2:0.600000
```

From the lines above we see the complete model for classification which uses RBF Gaussian kernels having width parameter $\sigma = 4$ and penalty parameter (upper bound on dual variables α_i) $C = 10$. The input is 2-dimensional and both inputs are scaled between -1 and 1 by using following scaling formula for the input $\mathbf{x}_i = [x_{i1} \ x_{i2}]^T$: $x_{i1s} = 0.0274 \ x_{i1} - 1.466$ and $x_{i2s} = 0.4x_{i2} - 1$. Finally, the model has a bias term $b = -0.051012$ and 4 support vectors (SVecs) with the weights and positions as follows:

1st SVec's weight is 10, it is placed at $[-0.39726 \ 1]^T$, (and thus, this is a **bounded** SVec),
 2nd SVec's weight is 4.570504, it is placed at $[-0.726027 \ -1]^T$, (thus, this is a **free** SVec),
 3rd SVec's weight is -8.951533, it is placed at $[-0.589041 \ 1]^T$, (thus, this is a **free** SVec),
 4th SVec's weight is -10, it is placed at $[-0.342466 \ 0.6]^T$, (thus, this is a **bounded** SVec).

Thus, the file SVC_opt_model.dat contains the data describing following SVM's model given the new input vector \mathbf{x}_{new} ,

$$y(\mathbf{x}_{new}) = 10g\left(\mathbf{x}_{new}, \begin{bmatrix} -0.397 \\ 1 \end{bmatrix}\right) + 4.571g\left(\mathbf{x}_{new}, \begin{bmatrix} -0.726 \\ -1 \end{bmatrix}\right) - 8.952g\left(\mathbf{x}_{new}, \begin{bmatrix} -0.589 \\ 1 \end{bmatrix}\right) - 10g\left(\mathbf{x}_{new}, \begin{bmatrix} -0.342 \\ 0.6 \end{bmatrix}\right) - 0.051$$

where $g(\mathbf{x}, \mathbf{c}_i) = e^{-\frac{1}{2}\left(\frac{\mathbf{x}-\mathbf{c}_i}{4}\right)^2}$, $i = 1, 4$ are the values of 4 selected Gaussian kernels (support vectors).

Because we are solving a classification problem the closeness of $y(\mathbf{x})$ to the label 1 or 2 will define the class belonging of an input vector \mathbf{x} .

REGRESSION

One can build a regression model from the best SVM stored in the file SVR_opt_model.dat in an absolutely same procedure as described for a classification above.

Note that in the case of regression the file containing the best model SVR_opt_model.dat will be of almost ‘identical’ structure as the file in classification. SVR_opt_model.dat looks as given below (note that instead of Gaussian kernels, now the polynomial ones have been used):

```
svm_type svr
kernel_type POLY
data_type Dense
degree 3.000000
C 100.000000
dim 12
scale_factor 0.022480 1.000142 0.020000 1.000000 0.073314 1.033724 4.115226 2.584362
0.383215 2.364629 0.020597 1.059732 0.181861 1.205430 0.086957 1.086957 0.003817
1.713740 0.212766 3.680851 0.005043 1.001614 0.055188 1.095475
epsilon 11.000000
bias 0.059121
nSV 24
0.473072 1:-0.991848 2:-0.560000 3:-0.604106 4:-0.810700 5:0.800345 6:-0.886715 7:0.414352
8:-0.478261 9:-0.454198 10:0.382979 11:1.000000 12:-0.900110
0.304813 1:-0.954954 2:-1.000000 3:0.401760 4:-0.094650 5:0.673884 6:0.921730 7:-0.833361
8:-0.652174 9:-0.175573 10:-0.553191 11:0.860810 12:-0.891280
-0.775058 1:-0.999076 2:-1.000000 3:-0.159091 4:-0.226337 5:-0.053842 6:0.604531 7:-
0.749868 8:-1.000000 9:-0.671756 10:0.787234 11:1.000000 12:-0.660596
...
...
...
```

Also note that the input is 12-dimensional as well as that we have shown only the first three SVs out of 24 support vectors obtained. The regression model is now,

$$y(\mathbf{x}_{new}) = \sum_{i=1}^{24} v_i K(\mathbf{x}_i, \mathbf{x}_{new}) + b = \sum_{i=1}^{24} v_i (\mathbf{x}_i^T \mathbf{x}_{new} + 1)^3 + b = v_1 (\mathbf{x}_1^T \mathbf{x}_{new} + 1)^3 + v_2 (\mathbf{x}_2^T \mathbf{x}_{new} + 1)^3 + v_3 (\mathbf{x}_3^T \mathbf{x}_{new} + 1)^3 + \dots + v_{24} (\mathbf{x}_{24}^T \mathbf{x}_{new} + 1)^3 + b$$

$$= 0.473072 \left[\begin{array}{cccccc} -0.991848 & -0.560000 & -0.604106 & \dots & -0.900110 & \dots \end{array} \right] \left[\begin{array}{c} x_{1new} \\ x_{2new} \\ x_{3new} \\ \vdots \\ x_{12new} \end{array} \right] + 1 \Big)^3 + \dots + 0.059121$$

Finally, note that the abbreviated form of the final SVM regression model is shown above. In particular, for a given new input (measurement, observation, sample) \mathbf{x}_{new} only the first (in)complete kernel value for the first support vector and \mathbf{x}_{new} is shown together with the bias at the end. (In fact, only first three components and the 12th one are shown in the row vector above). There are also 23 kernels (coming from the rest of 23 support vectors in between first kernel and bias) missing in expression for $y(\mathbf{x}_{new})$. However, they should be built and calculated in the same way as the first term shown.