SemiL (Copyright (**C**) 2004 Te-Ming Huang and Vojislav Kecman)

is efficient software for solving large-scale semi-supervised learning problem using graph based approaches. It can solve various semi-supervised learning problems as listed below:

Hard label approach with the maximization of smoothness, and
Soft label approach with the maximization of smoothness,

for all three types of models (i.e., Basic model, Norm Constrained Model and Bound Constrained Model) by using either Standard or Normalized Laplacian, e.g.,

with option *-l* 1 *–h* 1 *–mu* 0.0101 *-lambda* 0.0101, the following formulation is used.

$$\phi(F) = \frac{1}{2}\left(\sum_{i,j=1}^{n} W_{ij} \left\| \frac{1}{\sqrt{D_{ii}}} F_i - \frac{1}{\sqrt{D_{jj}}} F_j \right\|^2 + \lambda \sum_{i=l} \left\| F_i - Y_i \right\|^2 + \mu \sum_{j=u} \left\| F_j \right\|^2 \right)$$

It is important to note that *lambda* control the amount of penalty on the empirical error of the labeled point and *mu* controlled the norm of the output of the unlabeled points. It is the same as the consistency method proposed in [1] when alpha is equal to 0.99.

Please read the copyright file before using SemiL!

Installation

For Windows User
Unzip the file semil.zip and it will self-extract itself into the folder SemiL. The executable file "SemiL" will be in the folder "Windows". Rename the file "SemiL" to "SemiL.exe" to execute.
Once extracted, and after running MS DOS, the working directory in Command Prompt (MS DOS) must be c:/SemiL/Windows.

That means, once you are in Command Prompt type in:
cd c:/semil/Windows
The windows version of SemiL is developed in Visual C++ 6.0.
For Linux User
The executable file for Linux (SemiL) will be in .../SemiL/Linux folder.
The Linux version of SemiL is developed in KDvelop 2.1.3

This software can be installed with the use of Intel BLAS routine or without. The BLAS routine is highly recommended for the user whose problems are dense. For Windows users, the executable SemiL.exe in the "Windows" folder is Intel BLAS enabled, so you can straight away enjoy the high performance of the library. However, if you want to recompile the code, you need to purchase it from Intel. As for the Linux user, you probably have to purchase it from Intel and recompile the software. To compile the code with Intel BLAS enabled, just change the first line of the header file "dist_obj.h" from "#define intel_blas **1**" to #define intel_blas **0**".

Before using please **go through the options**. By typing
semil
all the SemiL routine's options will be displayed.

They are as follows:

-t Distance type
    1 = Euclidean distance
    2 = Cosine distance
-d Degree of the graph (this is a design parameter and for each problem the final model may have different degree)
-m Cachesize : set cache memory size in MB (default 0)
    (m is needed when working with dense raw data only, to speed up the calculations)
-l Standard or normalised Laplacian
    0 = standard Laplacian
    1 = normalized Laplacian
-h Hard or soft label
    0 = hard label
    1 = soft label
-k Kernel_type
    0 = RBF function exp(-(|u-v|^2)/gamma)\n
    1 = Polynomial function (not an option at the moment)
-p Degree of polynomial (at the moment not implemented)
-mu Penalty parameter valid for the Norm Constrained Model only.
-lambda Penalty parameter for the empirical error, valid for the Soft Label approach only.
-r Number of random experimental runs for a given setting
-g Gamma value for the RBF kernel (shape parameter of the n-dimensional Gaussian)
-pl Percentage of the labeled points in the experiment
-stp Precision for the solver (default = 1e-5)
-up_b Upper bound for the bounded constraint (default = inf)
-low_b Lower bound for the bounded constraint (default = -inf)
-nr Normalization of the output i.e., F* matrix
    0 = Without a Normalization
    1 = With a Normalization
-ocl One class labeling (Default 1)
    0 = Two Class labeling (-1 and +1)
    1 = One Class labeling(+1 only)

Input data format:

SemiL can take two different types of data as the input. For first time solving a given problem with SemiL, you need to convert your data set into the raw data file format as given below.

Raw data format:

<label1> <index1>:<value1> <index2>:<value2> ……

<label2> <index1>:<value1> <index2>:<value2> ……

<label1> is the desired label of the first data point and <label 2> is the desired label of the second data point. The <index1> is an integer value starting from one (1) and it tells to the program which dimension <value> belongs to. SemiL can take raw data in sparse form or in dense form. For data point *i* with unknown label, set the value of <label*i*> to zero.

Example: we have 7 4-dimensional measurements belonging to three classes and only one measurement per class is labeled. The data are given as:

	Dimension of the input			
Label	value 1	value 2	value 3	value 4
1	0	1.1	0.3	-1.1
0	-2	0	1.1	0.7
0	1.1	-3.1	0	1.1
2	0	0	0	2
3	5	-0.5	1	2.3
0	2	0	-4.1	0
0	0	1.1	0	3.7

Data in DENSE format are to be given as follows:

1 1:0 2:1.1 3:0.3 4:-1.1
0 1:-2 2:0 3:1.1 4:0.7
0 1:1.1 2:-3.1 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
0 1:2 2:0 3:-4.1 4:0
0 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
0 1:-2 3:1.1 4:0.7
0 1:1.1 2:-3.1 4:1.1
2 4:2
3 1:5 2:-0.5 3:1 4:2.3
0 1:2 3:-4.1
0 2:1.1 4:3.7

After solving the problem for the first time, SemiL will generate a distance matrix file (you should specify the name at the prompt) and a label file having the same name augmented by the label extension. You can use these two files during the design runs playing with various design parameters without an evaluation of a distance matrix each time.

In Windows version of SemiL, Intel BLAS is incorporated to improve the performance on evaluating the distance matrix when data is dense. You can specify the amount of cache by defining an option "-m ".

The program can be run in two modes - Experiment Mode or in Prediction One

Experiment Mode (ExM)
ExM tests different types of semi-supervised learning algorithms by inputting data set with all the data labeled. In this mode, it will randomly select a fixed number of data points as labeled points, and then it will try to predict the label for the rest of the points. By comparing the predicted labels and the true labels, the user can examine the performance of different settings for a semi-supervised learning. The number of data points to be selected is specified by option "-pl", which stands for percentage of data point to be labeled from all data. The user can specified how many experiments should be run by the option "-r ". To activate this mode, the user only needs to supply the routine with ALL the data labeled.

Predicting mode (PM)
The routine will run in PM as long as there is at least one label equal to zero. In the predicting mode, the program will predict the label of ALL the unlabeled data. To activate this mode, the user simply set the label of unlabeled points equal to 0 in the data file.

Getting started

1. Prepare your data in the format readable by the program. If your data is in Matlab, use the convtosp.m or convtode.m to convert it into Matlab variable. To use these routines, you need to put the label of your data points as **the first column** of your matlab variable in matlab. Convtosp.m will convert your full matlab variable into the proper format as a sparse input data. Convtoden.m will convert your full matlab variable into a dense input data for the program.
2. Once the data is prepared, you can use the command line to run the program. Below, we first run the problem **20 News Group Recreation** for which the data are extracted (by using the Rainbow software) and stored in the file **rec.txt** (in a sparse format).
3. To perform the run type in the following line in the directory of the exe file
   Semil -t 2 -d 10 -m 0 -l 0 -h 0 -k 0 -u 0 -g 10 -r 50 -pl 0.003 -lambda 0 -mu 0.5 rec.txt
4. Thus, the user starts with the raw data input to the program which will compute the distance matrix (used for the RBF model's only) and save it separately from the labels. It will produce a file named by us. Here we named it " rec2_10d.dat " for the output of the solver which will be saved as the file. Additionally, two more files will be created, namely "rec2_10d.dat.output" and "rec2_10d.dat.label". At the same time the error rate for each run will be recorded in the file 'error_rate.dat'.

Design stage:

5. After the distance matrix is calculated and associated with the corresponding labels (which are stored in separate files) a design by changing various model parameters (settings e.g., l, h, k, g, r, pl lambda, and mu) can start by typing in the following line.

   Semil -l 0 -h 0 -k 0 -u 0 -g 10 -r 50 -pl 0.003 -lambda 0 -mu 0 rec2_10d.dat rec2_10d.dat.label
   (or whatever name you called these two files it)
   The above line will implement Harmonic Gaussian method [2]. To use Global consistence model [1] use the following line.
   Semil **-l 1 -h 1** -k 0 -u 0 -g 10 -r 50 -pl 0.003 -lambda 0 -mu 0.5 rec2_10d.dat rec2_10d.dat.label

   In this setting, the computation of distances will be skipped and the program will read the distance matrix from file and use it for the simulation.
6. Same as in the run with raw data the results will be saved in three files:
   "rec2_10d.dat.output" , "rec2_10d.dat.label" and in "error_rate.dat".
   Also, the errors in each run will be printed on the screen.

References

1. Zhou, D., Bousquet, O., Lal, T. N., Weston, J., Schölkopf, B.: Learning with Local and Global Consistency, Advances in Neural Information Processing Systems **16**, (Eds.) Thrun, S., L. Saul and B. Schölkopf, MIT Press, Cambridge, Mass. (2004) 321-328
2. Zhu, X.-J., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using Gaussian fields and harmonic functions, Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, (2003)