Sparse Logistic Regression ToolBox ver1.1alpha

(Updated on 2009/06/05 by Okito Yamashita)

The Sparse Logistic Regression toolbox (**SLR toolbox** hereafter) is a suite of MATLAB functions for solving classification problems. It provides one of solutions for binary or multi-class classification problem. The unique feature is parameters of the classifier are learned in a sparse way, resulting in automatic feature selection while learning weight parameters in the classifier. This feature may be appropriate to classification problems of neuro-imaging data where only a limited number of training data (from several tens to several hundreds) can be used to classify a rather high dimensional feature vector (over one thousand). Furthermore SLR releases users from the time-consuming feature selection task preceding classification although results may be suboptimal.

~~~~~~

Installation

~~~~~~

SLR toolbox is a suite of MATLAB functions and scripts. MATLAB, a commercial engineering mathematics package, is required to use SLR toolbox. A couple of functions require the optimization toolbox (see below). Codes in the toolbox were written for MATLAB ver7.0.1 or later under UNIX. This toolbox has originally been developed by Okito Yamashita in ATR Computational Neuroscience laboratories for personal use.

To get installed the toolbox, you just download and unzip the file (SLR1.0beta.zip) wherever you like. You may also download two test-data acquired from two real experiments in order to see how binary and multi-class classification problems are solved (this is optional). Please start from demo functions ('demo_*.m') to learn how the functions in SLR toolbox work.

~~~~~~

Contents Summary (functions)

~~~~~~

Functions in SLR toolbox can be separated into 5 classes; low-level functions, run-level functions, demo functions, common functions, functions from others (see figure 1).

- “Low-level functions” are the functions that implement learning parameters in the classifiers. Most functions take a pair of label and a feature matrix as inputs and output estimated weight parameters.

- “Run-level functions” are the functions that implement whole procedure to solve a classification problem (i.e. normalization features → learning parameters → testing learning parameters). Most functions take pairs of label and a feature matrix for learning and testing as inputs and output estimated parameters as well as some performance measure. There are five functions for each of the binary classification problem and the multi-class classification problem (see below).
- “Demo functions” are for demonstration purpose. You can learn how functions in the toolbox work using some examples.
- “Common functions” are the functions that are commonly used in toolbox.
- “Functions from others” are the functions that are borrowed from EEGLAB toolbox and MATLAB File exchange.

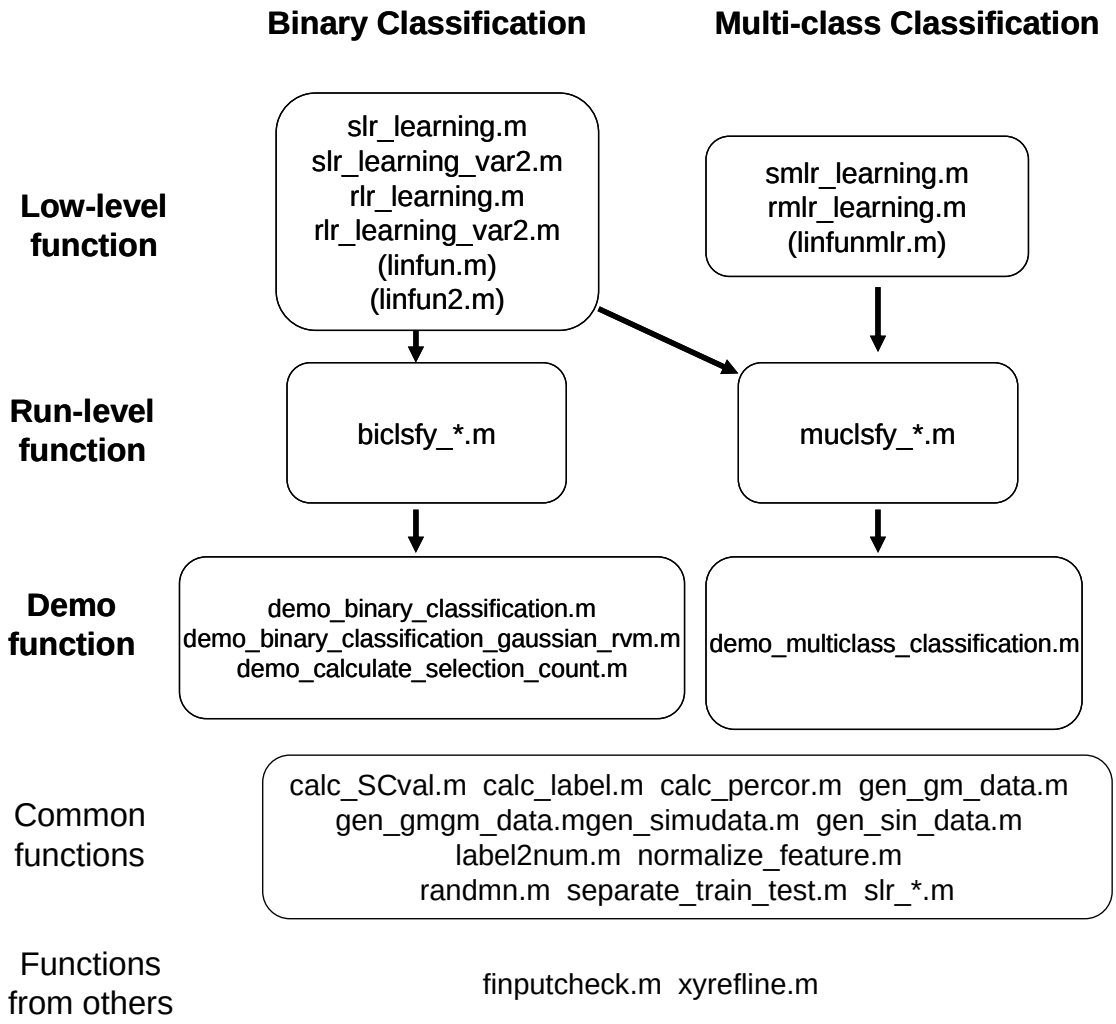

Figure 1: Functions in SLR toolbox

~~~~~~

Contents Details (Run-level functions)

~~~~~~

In "run-level functions", each of five functions of which name starts from 'biclsfy_*' implements five binary classifiers and each of five functions of of which name starts from 'muclsfy_*' implements five multi-class classifiers (see figure 2). Classifier in the row of 'Sparse' has feature selection property while classifiers in the row of 'Others' do not (for comparison purpose).

	Binary classification	**Multi-class classification**
Sparse	SLR-LAP (biclsfy_slrlap.m) SLR-VAR (biclsfy_slrvar.m)	SMLR (muclsfy_smlr.m) SLR-LAP-1vsR (muclsfy_slrlapovrm.m) SLR-VAR-1vsR (muclsfy_slrvarovrm.m) SLR-VAR-1vs1 (muclsfy_slrvarovo.m) New
Others	RLR-LAP (biclsfy_rlrlap.m) RLR-VAR (biclsfy_rlrvar.m) RVM (biclsfy_rvm.m)	RMLR (muclsfy_rmlr.m) RLR-VAR-1vsR (muclsfy_rlrvarovrm.m)

SLR-LAP = Sparse Logistic Regression (with Laplace approximation)
SLR-VAR = Sparse Logistic Regression (with variational approximation)
SMLR = Sparse Multinomial Logistic Regression

RLR-LAP = Regularized Logistic Regression (with Laplace approximation)
RLR-VAR = Regularized Logistic Regression (with variational approximation)
RMLR = Regularized Multinomial Logistic Regression

RVM = Relevance Vector Machine

1vsR = One-Versus-Rest
1vs1 = One-Versus-One

Figure 2: Supported classifiers in SLR toolbox (Run-level functions)

Binary classifiers

Two sparse classifiers and three non-sparse classifiers are supported so far.

- **SLR-LAP**: SLR with Laplace approximation. The marginal posterior-distribution of weight parameters is approximated by multivariate Gaussian distribution (see ref.[1] for details). This was developed for the research in ref.[1]. The optimization toolbox is required ('fminunc.m').
- **SLR-VAR**: SLR with variational approximation. The logistic function is approximated

by Gaussian distribution using a variational parameter (see ref.[2.3] or section 10.6 of ref.[4]). The faster and the less memory.

- **RLR-LAP**: Regularized logistic regression with Laplace approximation. This is not sparse algorithm. The regularization parameter is automatically determined by the algorithm. The optimization toolbox is required ('fminunc.m').
- **RLR-VAR**: Regularized logistic regression with variational approximation. This is not sparse algorithm. The regularization parameter is automatically determined by the algorithm.
- **RVM**: Relevance Vector Machine as proposed by Tipping (see ref.[5]). Bayesian version of Support Vector Machine (SVM). The linear and Gaussian kernels are supported. The Gaussian kernel RVM is only non-linear classifier supported in this toolbox.

Two sparse classifiers, SLR-LAP and SLR-VAR, are derived from the identical probabilistic model (see ref[1]) but different approximation to the posterior distribution. The difference between RLR-LAP and RLR-VAR is as in the same way. Among two sparse classification methods, I recommend use of SLR-VAR because it is faster and require less memory. It can be applied to higher dimensional problems with less computational resource.

Multi-class classifier

Three sparse classifiers and two non-sparse classifiers are supported so far.

- **SMLR :** Sparse Multinomial Logistic Regression (see ref.[1]). The multinomial distribution is used for observation. In general, memory and time required are huge. The optimization toolbox is required ('fminunc.m').
- **SLR-LAP-1vsR :** Combination of SLR-LAP classifiers. One-versus-the rest scheme is used. The optimization toolbox is required ('fminunc.m').
- **SLR-VAR-1vsR :** Combination of SLR-VAR classifiers. One-versus-the rest scheme is used. The faster computation and the less memory.
- **RMLR :** Regularized Multinomial Logistic Regression. One regularization parameter common to all the classes is automatically estimated. The optimization toolbox is required ('fminunc.m').
- **RLR-VAR-1vsR :** Combination of SLR-VAR classifiers. One-versus-the rest schemes is used. One regularization parameter per each class is automatically estimated.
- **SLR-VAR-1vs1**: Combination of SLR-VAR classifiers. One-versus-the one scheme is used. The faster computation and the less memory.

SMLR is a true multinomial classifier that uses multinomial distribution for observation. SLR-LAP-1vsR and SLR-VAR-1vsR are consisting of combination of sparse binary classifiers. Especially one-versus-the-rest scheme is employed (see chapter4 of ref[4] for example). In this release, one-versus-one scheme is also supported for SLR-VAR. In theory, SMLR is the best classifier for multi-class problem since the model learning take into account all the information among classes. But in my experience SLR-LAP-1vsR or SLR-VAR-1vsR, SLR-VAR-1vs1 do perform as well as SMLR probably due to a small number of training samples. Thus I recommend to start from SLR-VAR-1vs1 that is fast and requires less memory.

Details of 'biclsfy_slrvar.m'

In order to see what kinds of processing are done in each run-level function, the content of "biclsfy_slrvar.m" is explained. The other run-level functions share the same structure.

The function has following format;

```
[ww, ix_eff_all, errTable_tr, errTable_te, parm, AXall,Ptr,Pte] =…
  biclsfy_slrvar(x_train, t_train, x_test, t_test, varargin).
```

Input variables are a pair of a feature matrix and a label vector for training (x_train, t_train), and a pair of those for testing (x_test, t_test). Output variables are weight parameters (**ww**) that represents linear boundary between class1 and class2, results of classification (errTable_tr, errTable_te), probabilistic outputs (Ptr, Pte) and so on. 'varargin' takes 'key' -'value' form to specify optional parameters (as defined in a table of the function around line 75). The default values usually worked reasonably well.

In the function, the following steps are successively processed;

1. check optional variable
2. normalize feature matrix
3. add a bias regressor to the feature matrix depending on 'parm.usebias'
4. learning weight parameters (boundary parameters) of a classifier
5. evaluating percent correct for training and test data.

The step 2, normalization, is simply to apply linear transformation to each element in the

feature matrix so that all the elements range from -1 to 1. If ‘mean_mode’ = ‘each’ (default), the bias term of the linear transformation is calculated such that mean value of each feature gets zero (feature-wise). If ‘mean_mode’ = ‘all’, the bias term of the linear transformation is calculated such that mean value of all the elements gets zero (all elements mean). In the same way, ‘scale_mode’ defines the way to compute the slope of the linear transformation. If ‘norm_sep’ = 0 (default), the transformation derived from training data is applied to both training and test feature matrices. On the other hand, if ‘norm_sep’ = 1, two linear transformations derived from training and test data, respectively, are applied to training and test data.

:: Tips ::

- Among optional inputs, ‘scale_mode’ and ‘mean_mode’ may have most impact on performance. So you may try different parameters such as ‘all’ and ‘none’, if your result is not satisfactory. But it should be noted these parameters more often suffer from computational errors (scale difference).
- The option ‘invhessian’ is also important in some case. It does not affect performance but computation time. If the dimension of your feature matrix is larger than the number of training samples, the default setting is fast. Otherwise, setting invhessian = 1 is fast.

~~~~~~

Brief Mathematical Basics

~~~~~~

Logistic regression (LR) is a well-known classifier originally developed in statistics. SLR is a Bayesian extension of LR in which a sparseness prior is imposed on LR. In the literature, two kinds of the sparsenss prior has been suggested; Automatic Relevance Determination (ARD) prior ([7,8]) and Laplace prior ([5]). In this toolbox the ARD prior is employed. Please see the appendix of [1] for equations of the model and derivation of the algorithm. It should be noted that for SLR having feature selection property, the boundary must be linear (but not using linear kernel). This is because in this case each feature has its own weight parameter and thereby sparse estimation of weight parameters can be interpreted as removing irrelevant features.

~~~~~~

Referencing the toolbox

~~~~~~

When using this tool for a paper please refer to the following paper:

Yamashita O, Sato MA, Yoshioka T, Tong F, Kamitani Y (2008).
Sparse estimation automatically selects voxels relevant for the decoding of fMRI activity patterns. *Neuroimage*. Oct 1;42(4):1414-29.

The above manuscript contains basics of SLR (SLR-LAP) and applications to fMRI decoding.

~~~~~~

Feedback & Bug report

~~~~~~

Any feedback and bug report are welcome. Please keep contact with me (oyamashi@atr.jp). I would like to respond as quickly as possible.

~~~~~~

Licencing & Copy Right

~~~~~~

Acknowledgements

This toolbox is brought to you by ATR Computational Neuroscience laboratories in Kyoto. This research was supported in part by the NICT, Honda Research Institute, the SCOPE, SOUMU, the Nissan Science Foundation, and grants from the National Eye Institute to FT (R01 EY017082 and R01 EY14202).

References

---

[1] Yamashita O, Sato MA, Yoshioka T, Tong F, Kamitani Y (2008).
Sparse estimation automatically selects voxels relevant for the decoding of fMRI activity patterns. *Neuroimage*. Oct 1;42(4):1414-29.

[2]Jaakkola TS, Jordan MI (2000), Bayesian parameter estimation via variational methods, *Statistics and Computing*, 10, pp.25--37

[3] Bishop C, Tipping ME (2000),Variational relevance vector machines, *Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence*, pp.46-53

[4] Bishop C (2006), Pattern recognition and machine learning, Springer, New York

[5] Tipping ME(2001), Sparse Bayesian Learning and the Relevance Vector Machine, *J Machine Learning Research*, 1, pp.211-244

[6] Krishnapuram B, Carin L, Figueiredo MAT, and Hartemink AJ (2005), Sparse Multinomial Logistic Regression Fast Algorithms and Generalization Bounds, IEEE Trans. Pattern Analysis and Machine Intelligence, 27, pp.957-968

[7] MacKay D (1992), Bayesian interpolation, Neural Computation, 4, pp.415-447

[8] Neal RM(1996), Bayesian Learning for Neural Networks, Lecture Notes in Statistics 118,. Springer