

A no-nonsense beginner's tool for GMLVQ Version 2-2

(for a list of changes: see end of document)

Getting started

Start Matlab in the directory containing the unzipped archive. Provide a labelled data set in the form

fvec: an array of size (**nvf**, **ndim**) containing the set of **nvf** feature vectors in **ndim** dimensions.
lbl: the corresponding vector of **nvf** class labels. It is required to use labels 1,2,3,...

e.g. by loading one of the sample data sets, which are used in the example demos:

iris.mat iris-small.mat uci-segmentation-sampled.mat
wine-hi-lo.mat twoclass-difficult.mat twoclass-simple.mat twoclass-simple-small.mat

Most parameters of the training procedure are set in the following function, see commands marked in green,
see comments for explanation and recommended settings.

```
function [showplots,doztr,mode,rndinit,etam,etap,mu,decfac,incfac,ncop] = set_parameters(fvec)
% set general parameters
% set initial step sizes and control parameters of
% modified procedure based on [Papari, Bunte, Biehl]
nvf=size(fvec,1); ndim = size(fvec,2);

% GMLVQ parameters, explained below
showplots = 1;
doztr      = 1;
mode       = 1;
rndinit    = 0;
mu         = 0;

% showplots (0 or 1): plot learning curves etc?                recommended: 1
% doztr (0 or 1): perform z-score transformation based on training set recommended: 1
% mode
% 0 for matrix without null-space correction
% 1 for matrix with null-space correction                      (recommended)
% 2 for diagonal matrix (GRLVQ)                               (discouraged)
% 3 for GLVQ with Euclidean distance (equivalent)
% rndinit
% 0 for initialization of relevances as identity matrix        (recommended for first experiments)
% 1 for randomized initialization of relevance matrix
% mu
% control parameter of penalty term for singularity of Lambda
% mu=0: unmodified GMLVQ                                     (recommended for first experiments)
% mu>0: prevents singular Lambda
% mu very large: Lambda proportional to Identity (Euclidean)

% parameters of stepsize adaptation
if (mode<2); % full matrix updates with (0) or w/o (1) null space correction
    etam = 2; etap = 1; % recommended example values: etam=2, etap=1
elseif (mode==2) % diagonal relevances only (discouraged)
    etam = 0.2; etap = 0.1; % recommended example values: etam=0.2, etap=0.1
elseif (mode==3) % GLVQ, equivalent to Euclidean distance
    etam=0; etap = 1;
end;
decfac = 1.5; % step size factor (decrease) for Papari steps example value: decfac=1.5
incfac = 1.1; % step size factor (increase) for all steps example value: incfac=1.1
ncop = 5; % number of waypoints stored and averaged example value: ncop = 5
end
```

If not specified otherwise, in the following, all parameters are set as given above (recommended values)

A) Single runs of GMLVQ, visualization and classification Inspection of the data set, convergence behavior, etc.

`[gmlvq_system, training_curves, param_set]= run_single(fvec, lbl, totalsteps, plbl)`

Performs a single training process for the entire data set. Cost function (divided by number of examples), error rates and AUROC as functions of the number of batch gradient training steps are saved in a structure `training_curves` and the final LVQ system after `totalsteps` gradient descent steps is saved in structure `gmlvq_system`. The structure `param_set` contains all system parameters.

If `showplots==1`, training curves (errors, AUROC, cost function, step sizes) are plotted and the LVQ system is visualized in terms of prototype vectors and relevance matrix; in addition, the ROC of the final system displayed and the data set is visualized in terms of the leading two eigenvectors of the relevance matrix. The vectors of feature means and standard deviations are also returned.

The obtained system can be applied to 'novel' data in terms of the function

`[crisp,score,margin,costf]= classify_gmlvq(gmlvq_system,fvec,doztr,lbl)`

Classifies the feature vectors in `fvec` using the GMLVQ configuration stored in `gmlvq_system` (e.g. output of `run_single`, see above). If `lbl` is specified, performance of the classifier is evaluated in terms of `margin` and the test equivalent of the GLVQ cost function `costf`. Otherwise, only output variables `crisp` (discrete labels assigned by the classifier) and `score` (corresponding score between 0 and 1) are meaningful.

Demo I: a seven class data set, single run with one prototype per class (UCI segmentation data set, 1 trivial dimension removed, 800 examples randomly selected)

```
>> load uci-segmentation-sampled.mat
>> [gmlvq_system,curves_single,param_set]=run_single(fvec,lbl,50);
```

default: one prototype per class

prototype configuration plbl = 1 2 3 4 5 6 7

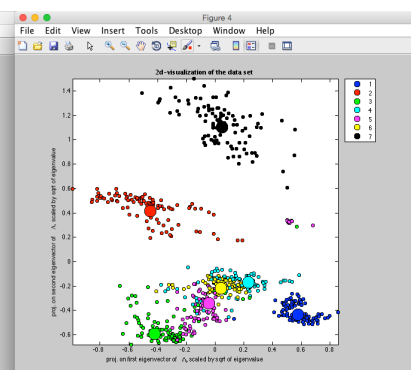
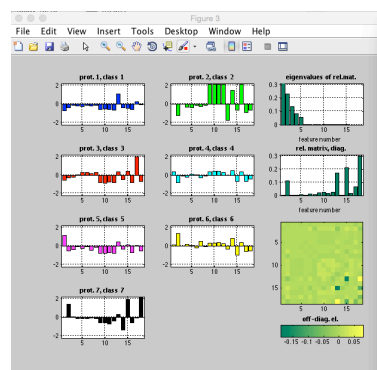
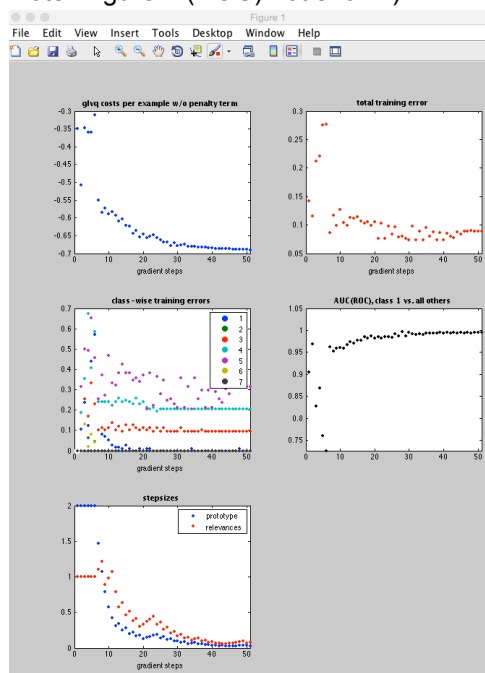
number of training steps totalsteps = 50

matrix relevances with null-space correction

Warning: multi-class problem, ROC analysis is for class 1 (neg.) vs. all others (pos.)

minimum standard deviation of features: 0.02525

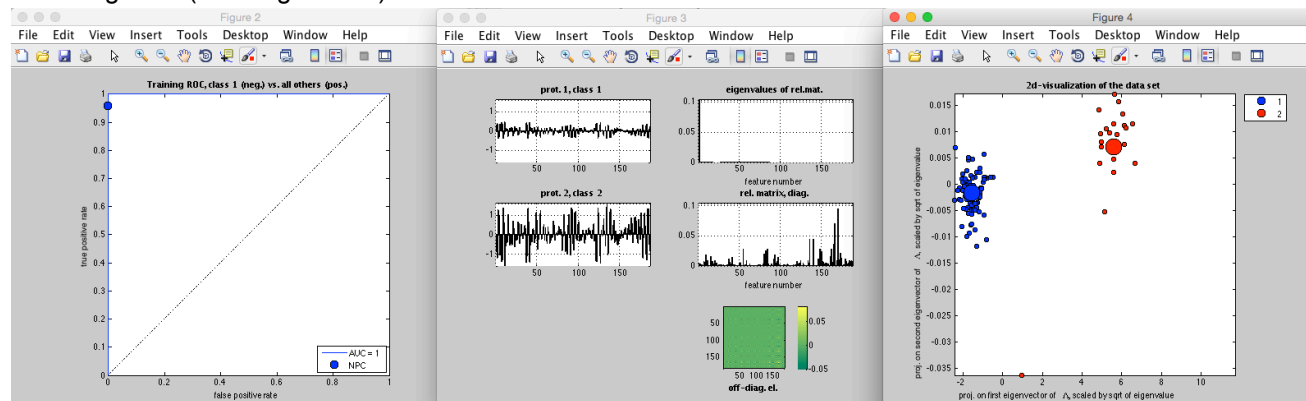
Note: Figure 2 (ROC) not shown)



Demo II: a simple two-class problems, 186-dim. feature vectors, 110 samples

```
>> load twoclass-simple
>> [gmlvq_system,curves_single,param_set]=run_single(fvec,lbl,50);
default: one prototype per class
prototype configuration    plbl = 1    2
number of training steps    totalsteps = 50
matrix relevances with null-space correction
Warning: vector lbl has been transposed
minimum standard deviation of features: 0.0090926
```

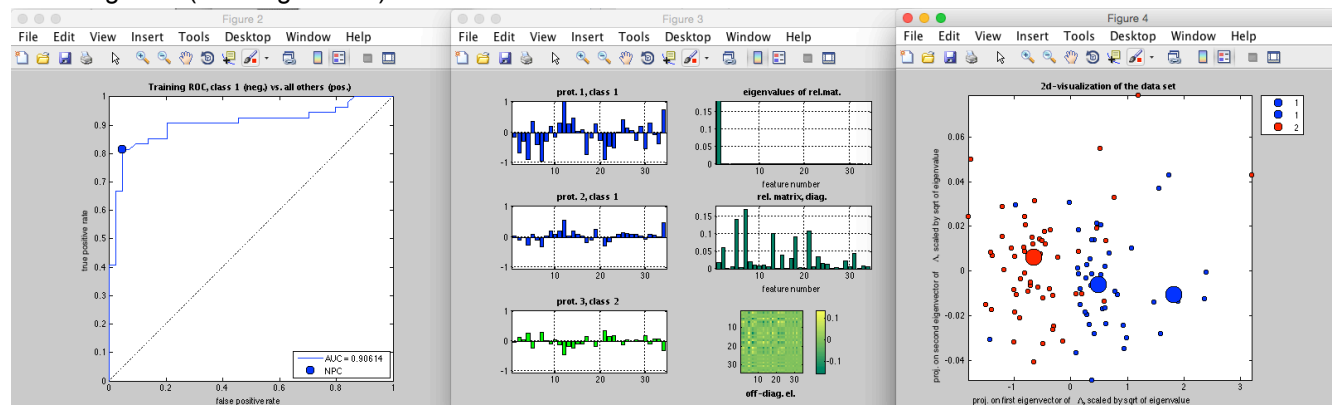
Note: Figure 1 (learning curves) not shown



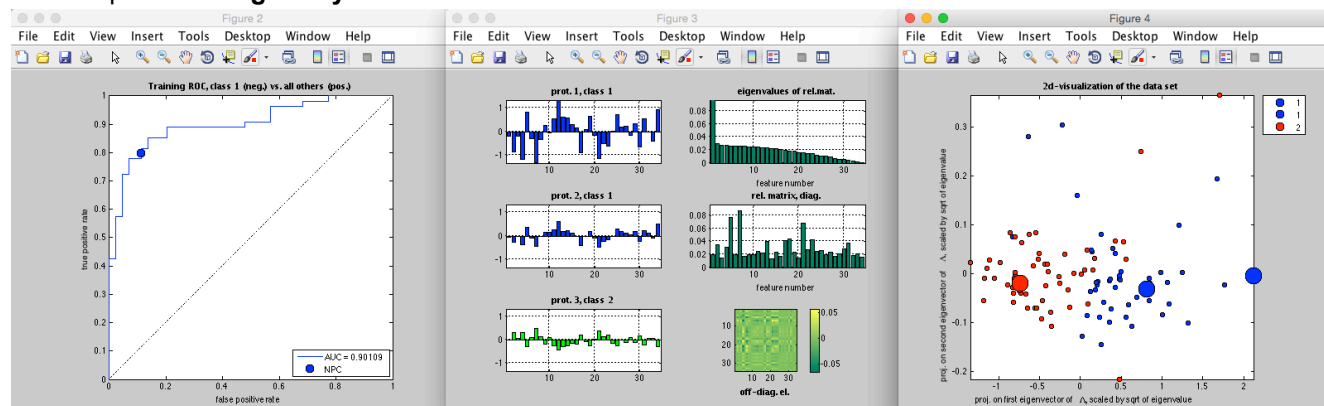
Demo III: a difficult two-class problems, 32-dim. feature vectors, 98 samples

```
>> load twoclass-difficult
>> [gmlvq_system,curves_single,param_set]=run_single(fvec,lbl,50,[1 1 2]);
prototype configuration    plbl = 1    1    2
number of training steps    totalsteps = 50
matrix relevances with null-space correction
minimum standard deviation of features: 0.56657
```

Note: Figure 1 (learning curves) not shown



For comparison: singularity control with $\mu = 0.2$:



B) Learning curves analysis and validation of the GMLVQ classifier

Repeated random selection of validation samples

```
[gmlvq_mean, roc_validation, lcurves_mean, lcurves_sdt, param_set ] = ...  
run_validation(fvec, lbl, totalsteps, nruns, prctg, plbl);
```

Performs `nruns` of training where `prctg` % of the data (randomly selected) are used for testing while the rest is used for training. The LVQ system (prototypes, matrix) is determined in each run, but only the mean configuration (over the validation runs) and the corresponding standard deviations are saved and displayed (if `showplots=1`). Analogously, mean learning curves (and standard deviations) with respect to training and validation sets are computed and potentially displayed. The threshold averaged test set ROC as well as the confusion matrix of the Nearest Prototype Classifier are computed, output and displayed for the final GMLVQ configuration.

Leave-One-Out validation, recommended for very small data sets only (Note: L1O can seriously mis-estimate performance)

```
[gmlvq_mean, roc_l1O, lcurves_mean, lcurves_sdt, param_set]= run_l1O(fvec, lbl, totalsteps, plbl);
```

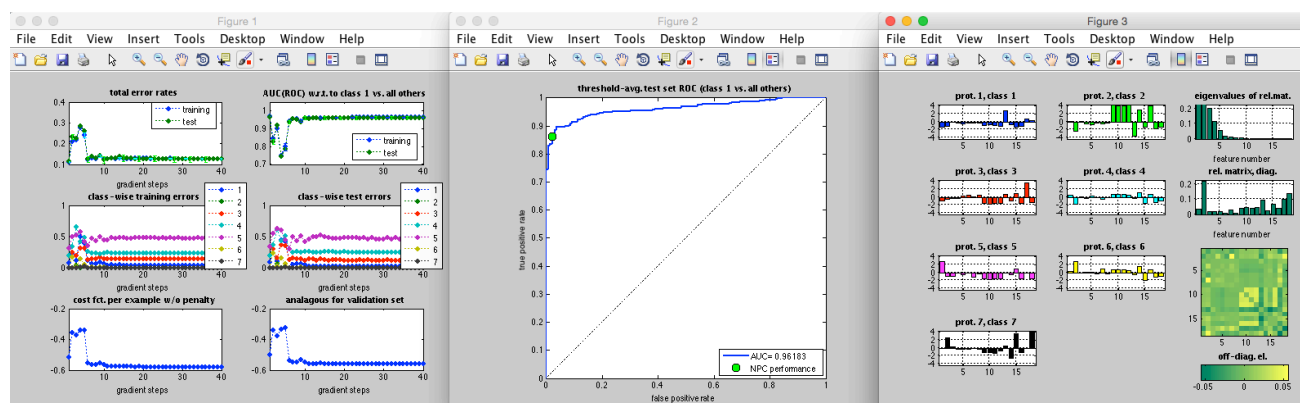
Performs a Leave-One-Out evaluation of the classifier. In each run, one example is excluded from training and used for testing. The LVQ system (prototypes, matrix) is determined in each run, but only the mean configuration (over the L1O runs) and the corresponding standard deviations are saved and displayed (if `showplots=1`). The numerical and graphical output is analogous to `run_validation`, however `run_l1O` does not compute learning curves in terms of test error and only a single test set ROC and NPC confusion matrix are determined at the end of training for all left out samples.

Demo IV: reduced UCI segmentation data set (see demo III)

validation by repeated runs with 10% randomly selected test data (random matrix initialization)

```
>> load uci-segmentation-sampled  
>> [gmlvq_mean,roc_val,lcurves_mean,lcurves_std,param_set]=run_validation(fvec,lbl,40,10,10)  
matrix relevances with null-space correction  
default: one prototype per class  
prototype configuration plbl = 1 2 3 4 5 6 7  
learning curves, averages over 10 validation runs  
with 10 % of examples left out for testing  
Warning: multi-class problem, ROC analysis is for class 1 (neg.) vs. all others (pos.)  
minimum standard deviation of features: 0.025258
```

(note: legends shifted manually in Figure 1)



Example output: averaged confusion matrix (Nearest prototype classifier, given as percentages)

```
>> roc_val.confmat
```

```
ans =
```

```
96.2984    0    3.0565    0    0.6451    0    0
    0 100.0000    0    0    0    0    0
 4.0174    0   93.5162    0   2.0924   0.3740    0
 4.1374    0    2.3858  73.5697   8.4987  11.4085    0
23.9696    0   22.0283   4.9418  49.0603    0    0
    0    0    0    0    0 100.0000    0
    0    0    0    0    0    0 100.0
```

row index: true class label of test sample

column index: class label predicted by GMLVQ system

Demo V: a difficult two-class data set (see demo III)

validation by repeated runs with 10% randomly selected test data (random matrix initialization)

```
>> load twoclass-difficult.mat
```

```
>>[gmlvq_mean,roc_val,lcurves_mean,lcurves_std,param_set]=run_validation(fvec,lbl,50, 10,10)
```

matrix relevances with null-space correction

default: one prototype per class

prototype configuration plbl = 1 2

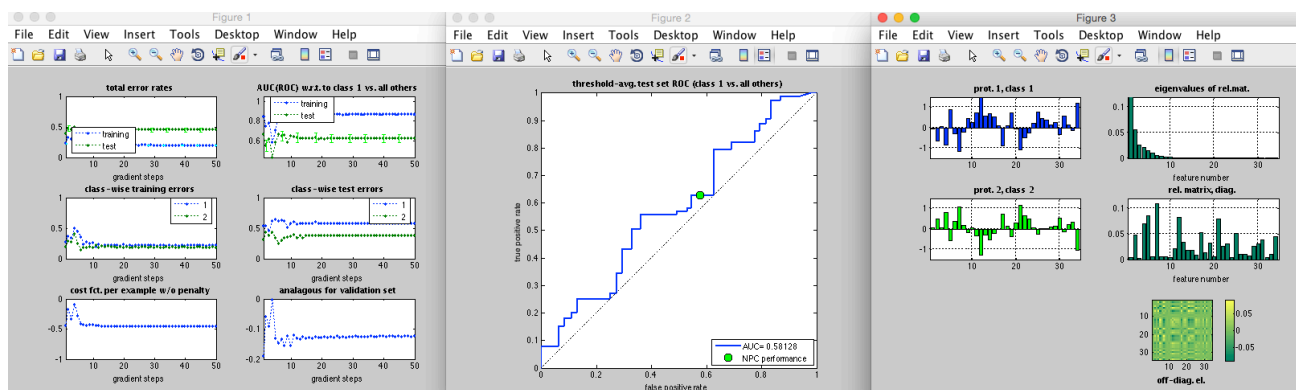
learning curves, averages over 10 validation runs

with 10 % of examples left out for testing

minimum standard deviation of features: 0.56657

....

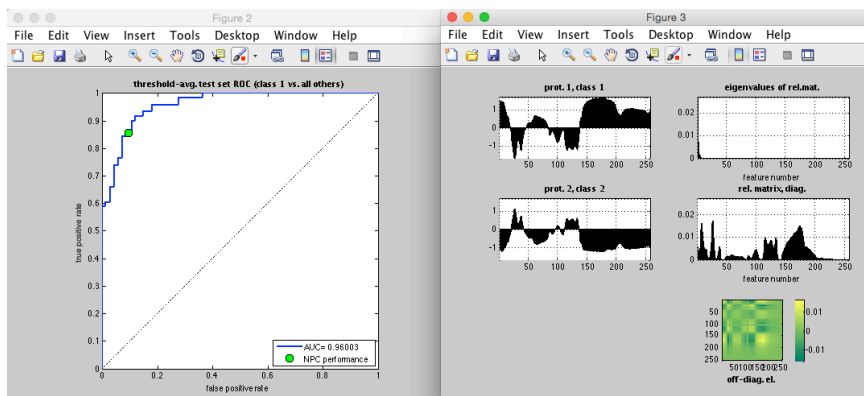
Poor validation performance compared to training set accuracies (III):



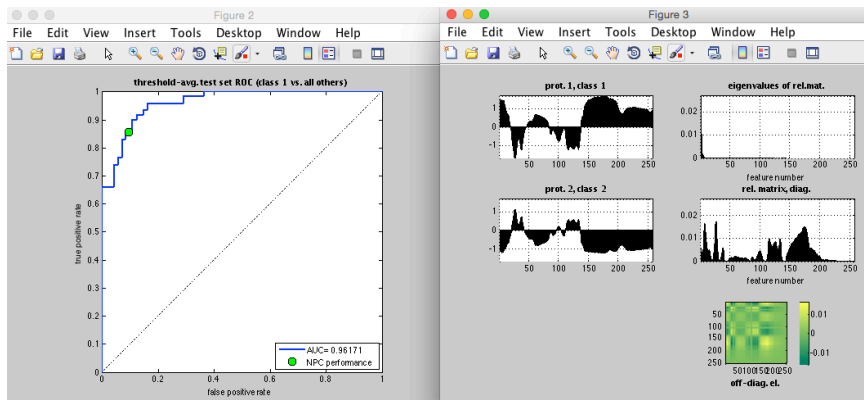
Demo VI: UCI wine data set, classification according to high/low alcohol content (threshold arbitrarily set at median alcohol content in the data set)

```
>> load wine-hi-lo
>> [gmlvq_mean,roc_val,lcurves_mean,lcurves_std,param_set]=run_validation(fvec,lbl,50,10,10)
matrix relevances with null-space correction
default: one prototype per class
prototype configuration      plbl = 1 2
learning curves, averages over 10 validation runs
with 10 % of examples left out for testing
minimum standard deviation of features: 0.003421
....
```

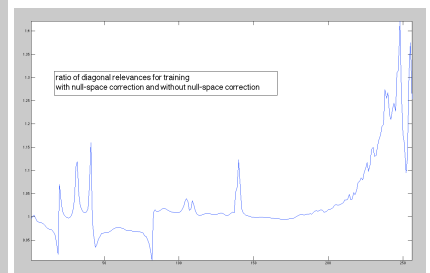
mode=0 (no null-space correction)



mode=1 (with null-space correction)



ratio of diag. relevances with and without null-space correction:

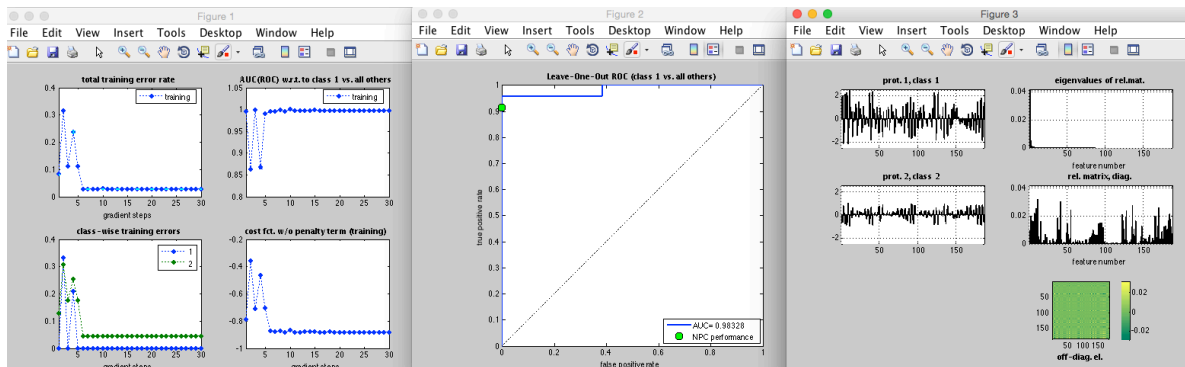


Note: null-space correction yields slightly better generalization (validation) behavior

Demo VII: a small subset of the simple two-class data (see demo II)

The last resort: **Leave-One-Out validation**

```
>> load twoclass-simple-small
>> [gmlvq_mean,roc_val,lcurves_mean,lcurves_std,param_set]=run_l1O(fvec,lbl,30)
default: 1 prototype per class
prototype configuration plbl = 1 2
matrix relevances with null-space correction
learning curves, averages over 36 l1O runs
Warning: vector lbl has been transposed
minimum standard deviation of features: 0.0089232
leave one out: 1 of 36
matrix relevances with null-space correction
...
```



Appendix: List of input arguments and parameters:

- fvec:** array of size (**nvf**, **ndim**) containing the set of **nvf** feature vectors in **ndim** dimensions.
- lbl:** the corresponding vector of **nvf** class labels. It is required to use labels 1,2,3,...
- totalsteps:** number of batch gradient steps to be performed in each training run
- nruns:** number of validation runs (random splits of the data set) in **run_validation**
- prctg:** percentage of test examples left out for each run in **run_validation** (rounded in practice)
- plbl:** (default: 1 per class label) labels assigned to the prototypes, this also specifies the number of prototypes per class, e.g. **plbl=[1,1,2,2,3]** for in total 5 prototypes with class labels 1,2,3 according to the list.
It is recommended to use the default first and add prototypes for seemingly difficult classes, subsequently, or determine the prototype configuration by (nested) validation
- showplots:** if set to 1, key results are displayed graphically (see above)
- doztr:** if set to 1, a standard z-score-transformation based on the respective training set is performed and applied analogously to the validation set data
- mode:** control LVQ version
mode=0 matrix without null space correction
mode=1 matrix with null--space correction (recommended by default)
mode=2 diagonal matrix (GRLVQ)
Note: this choice is not recommended as GRLVQ is more sensitive to the precise setting of learning rates and training times, in general. GMLVQ (mode 1 or 2) has proven much more robust in practice.
mode=3 GLVQ with Euclidean distance (relevance matrix proportional to identity)
- rndinit:** if set to 1, the relevance matrix is initialized randomly (if applicable), otherwise it is proportional to the identity matrix, initially (recommended initially)
- mu:** Lagrange parameter of singularity control
mu=0 unmodified GMLVQ algorithm (recommended for initial experiments)
mu>0 non-singular relevance matrix is enforced, mu controls dominance of leading eigenvectors continuously
-

Most of the above and a number of additional parameters concerning the gradient descent procedure and stepsize adaptation are set in function "**set_parameters**", see above.

Pre-defined values should work okay for an initial exploration of the data set. Performance can be optimized by tuning initial step sizes and adapting the number of gradient steps.

Key References: (see www.cs.rug.nl/biehl for pre/re-prints)

Introduction of Generalized Matrix Relevance LVQ (there: stochastic gradient descent)

P. Schneider, M. Biehl, B. Hammer,

Adaptive Relevance Matrices in Learning Vector Quantization

Neural Computation 21: 3532-3561 (2009)

Singularity control by penalty term: (a slightly misleading use of the term “regularization”)

P. Schneider, K. Bunte, H. Stiekema, B. Hammer, T. Villmann, M. Biehl

Regularization in Matrix Relevance Learning

IEEE Trans. Neural Networks 21: 831-840 (2010)

Null-space projection: (+extensions not implemented here, more appropriate use of “regularization”)

M. Strickert, B. Hammer, T. Villmann, M. Biehl

Regularization and improved interpretation of linear data mappings and adaptive distance measures, 2013 IEEE Symp. on Computational Intelligence and Data Mining (CIDM)

In: Proc. IEEE SSCI 2013

Step-size control: (here extended to matrix and prototypes treated separately)

G. Papari, K. Bunte, M. Biehl

Waypoint averaging and step size control in learning by gradient descent

Technical Report, In: MIWOCI 2011, Mittweida Workshop on Computational Intelligence, Machine Learning Reports MLR-2011-06: 16-26 (2011) [attached as pdf]

Main changes of version 2 compared to version 1 (as of April 2015)

- less explicit input parameters to functions (moved to function [set_parameters](#))
- improved step size control, really independent for matrix and prototype updates
- singularity control via penalty term included, parameter [mu](#) introduced
- [single_run](#) also displays temporal evolution of step sizes
- null-space correction controlled in [set_parameters](#) (no dimension-dependent default anymore)
- initial step sizes independent of dimension and/or number of examples

Main changes of version 2.1 compared to version 2 (as of August 23, 2015)

- corrected legends in plots showing step size vs. learning time
- corrected calculation of mean confusion matrix in `run_validation`. Now the confusion matrix is determined in each validation run separately (in terms of percentages) and then averaged over validation runs in the end. Resulting matrix is now in line with the averaged class-wise errors.

Change in version 2.2 compared to version 2.1 (April 2016)

- ROC calculation has been modified, it is now based on differences $d(x, w_1) - d(x, w_2)$ without normalization by $d(x, w_1) + d(x, w_2)$. See "compute_roc.m" for comments and details.