



g. *CLASSIFYtoolbox*

advanced biosignal processing and analysis

<http://www.gtec.at>
office@gtec.at

VERSION 5.16.02

USER MANUAL

Copyright 2017 g.tec medical engineering GmbH

CONTENT:

- PREFACE 4**
 - REQUIRED PRODUCTS..... 5
 - USING THIS GUIDE 6
 - CONVENTIONS 7
- HARDWARE AND SOFTWARE REQUIREMENTS 8**
- GENERATING A FEATURE MATRIX 9**
 - TRIAL ATTRIBUTES 12
 - TIME POINTS 15
 - TIME SEGMENTS 18
- GENERATING A CLASSIFIER..... 26**
 - LINEAR CLASSIFIER..... 28
 - Multi-Class LDA* 28
 - Minimum Distance Classifier*..... 30
 - NEURAL NETWORK 43
 - DSL VQ..... 47
 - SUPPORT VECTOR MACHINE CLASSIFIER 50
 - COMPARISON OF LDA AND DSL VQ 55
- RECEIVER OPERATOR CURVE 62**
- DSL VQ FEATURE WEIGHTING 65**
- KMEANS CLUSTERING 69**
- USING THE CLASSIFIER..... 71**
 - APPLY CLASSIFIER 72
 - TEST CLASSIFIER 75
 - CLASSIFICATION OUTPUT MAPPING 78
- DATA ACCESS..... 83**
 - USING THE GET COMMAND..... 83
 - USING THE SET COMMAND 83
 - ACCESSING THE FEATURE MATRIX 84
 - ACCESSING THE CLASSIFIER OBJECT 85
- HELP 86**
- BATCH MODE 87**
- PRODUCT PAGE..... 89**

To the Reader

Welcome to g.tec's world of medical and electrical engineering!

Discover the only professional biomedical signal processing platform under MATLAB and Simulink. Your ingenuity finds the appropriate tools in the g.tec elements and systems. Choose and combine flexibly the elements for biosignal amplification, signal processing and stimulation to perform even real-time feedback.

Our team is prepared to find the better solution for your needs.

Take advantage of our experience!

Dr. Christoph Guger

Dr. Guenter Edlinger

Researcher and Developer

Reduce development time for sophisticated real-time applications from month to hours. Integrate g.tec's open platform seamlessly into your processing system. g.tec's rapid prototyping environment encourages your creativity.

Scientist

Open new research fields with amazing feedback experiments. Process your EEG/ECG/EMG/EOG data with g.tec's biosignal analyzing tools. Concentrate on your core problems when relying on g.tec's new software features like ICA, AAR or online Hjorth's source derivation.

Study design and data analysis

You are planning an experimental study in the field of brain or life sciences? We can offer consultation in experimental planning, hardware and software selection and can even do the measurements for you. If you have already collected EEG/ECG/EMG/EOG, g.tec can analyze the data starting from artifact control, do feature extraction and prepare the results ready for publication.

Preface

This section includes the following topics:

[Required Products](#)

[Using This Guide](#) - Suggestions for reading the handbook

[Conventions](#) - Text formats in the handbook

Required Products

g®.CLASSIFYtoolbox uses:

g®.BSanalyze – the advanced biosignal analysis software package from g.tec

MATLAB – as basic matrix operation platform

Signal Processing Toolbox - to give access to standard signal analysis tools

Using This Guide

“[Generating a Feature Matrix](#)” shows how to extract specific features that are used for the classification task.

Chapter “[Generating a Classifier](#)” demonstrates the set-up of linear and non-linear classifiers and shows how to classify the feature matrix. Linear Discriminant Analysis (LDA), Minimum Distance Classifier (MDC), Multi-layer Perceptron (MLP), Radial Basis Function (RBF) and Distinction Sensitive Learning Vector Quantization (DSLQ) are explained.

“[Receiver Operator Curve](#)” demonstrates the computation of the sensitivity and specificity and presentation as ROC curve.

Chapter “[KMEANS Clustering](#)” demonstrates the clustering on a three-class problem.

“[DSLQ Feature Weighting](#)” explains the necessary steps for feature selection based on the DSLQ algorithm.

“[Using the Classifier](#)” explains how to apply already calculated classifiers on new data and how to test a classifier on new data.

“[Data Access](#)” shows how to access the feature matrix and classifier objects from the MATLAB command line.

Chapter “[Help](#)” explains the usage of the on-line help, the printable documentation and the function help.

Chapter “[Batch-Mode](#)” shows how to use the g.BSanalyze commands from the MATLAB command line.

Conventions

Item	Format	Example
MATLAB code	Courier	to start simulink, type simulink
String variables	<i>Courier italics</i>	set(P_C, 'PropertyName', ...)
Menu items	Boldface	Select Save from the File menu.

Hardware and Software Requirements

For Hardware and Software Requirements see the g.BSanalyze manual.

Generating a Feature Matrix

The first step in generating a feature matrix is to calculate specific features (e.g. bandpower values) with the Parameter Extraction methods of g.BSanalyze. There are three options available:

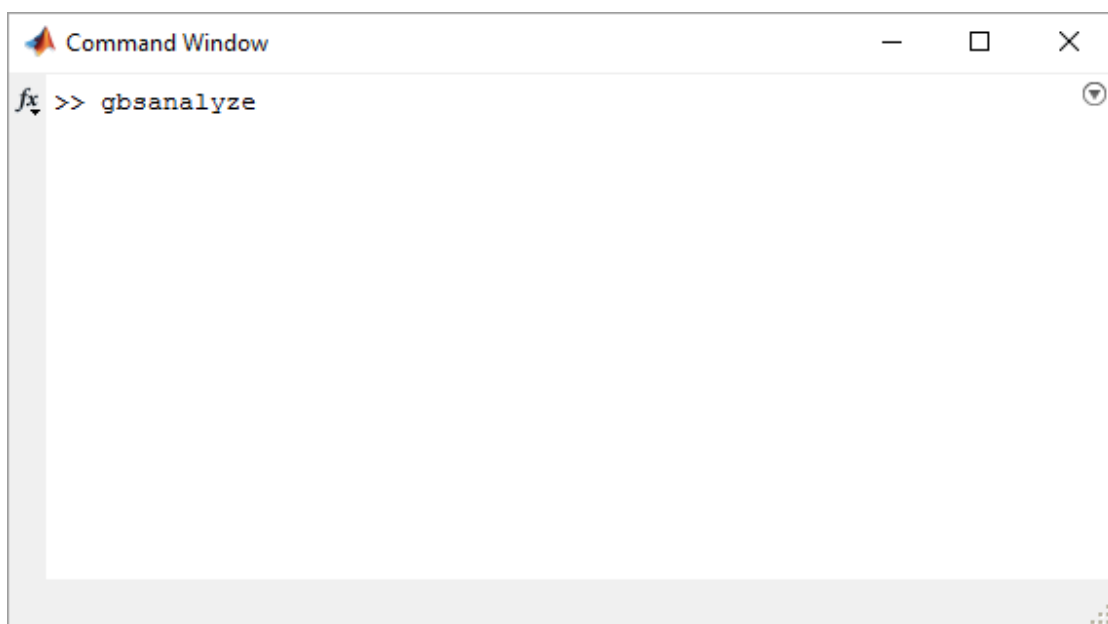
- Trial attributes - extract the features of trials with specific attributes
- Time points - extract the features of the signals at specific time points
- Time segments - extract the features of the signals of specific segments

Perform the following steps:

1. After starting MATLAB and setting the correct path, type:

```
gbsanalyze
```

into the MATLAB command line



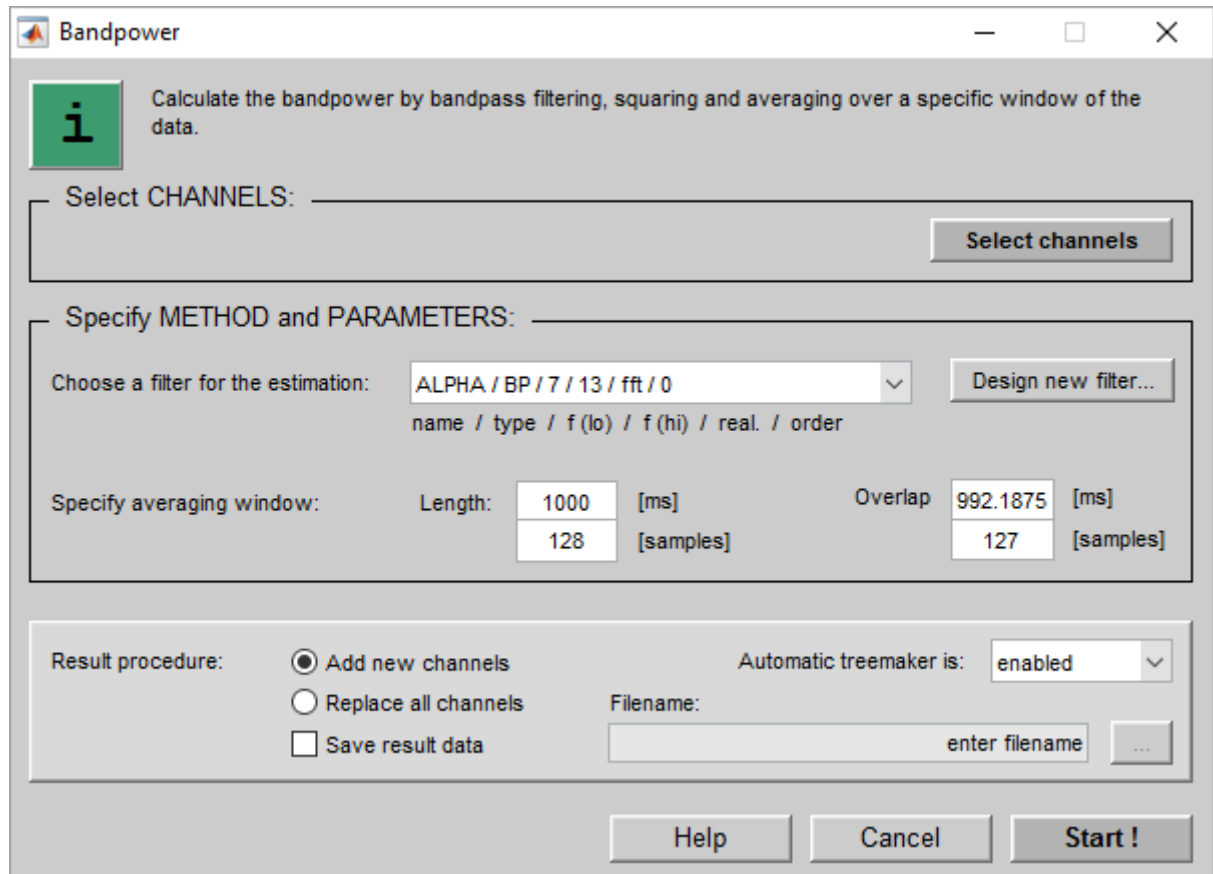
g.BSanalyze starts with a blank data window

2. Select **Load Data** under the **File** menu and open the file `session1234triggered.mat` from the following directory:

```
Documents\gtec\gBSanalyze\testdata\BCI
```

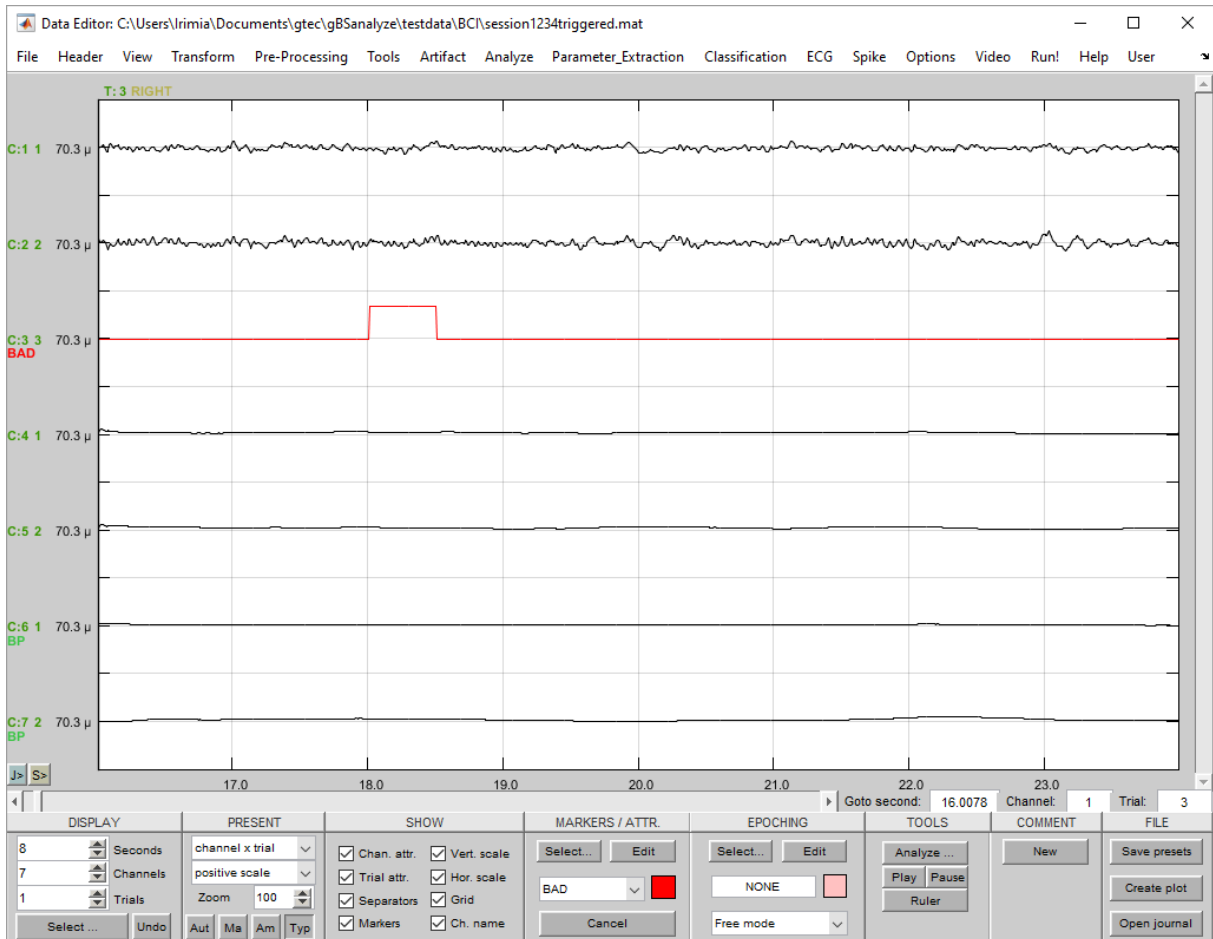
The Data Editor shows a brain-computer interface (BCI) experiment data-set with 2 EEG channels and 1 trigger channel. The first channel was recorded from channel C3, the second channel from channel C4. The paradigm is described in detail in the g.BSanalyze documentation in chapter **Data-sets – Movement Imagination**.

3. To extract relevant information out of the raw EEG data select **Bandpower** under the **Parameter Extraction** menu. Select the **ALPHA** filter to calculate the bandpower between 7 and 13 Hz and set the **Length** of the averaging window to 128 samples with an **Overlap** of 127 samples.



4. Press the **Select channels** button and select only the two EEG channels 1 and 2 for the operation
5. Chose **Add new channels** to append the bandpower values to the raw data
6. Press **Start** to perform the operation
7. Repeat steps 4 to 7 with the **BETA-3** filter (14 to 20 Hz)

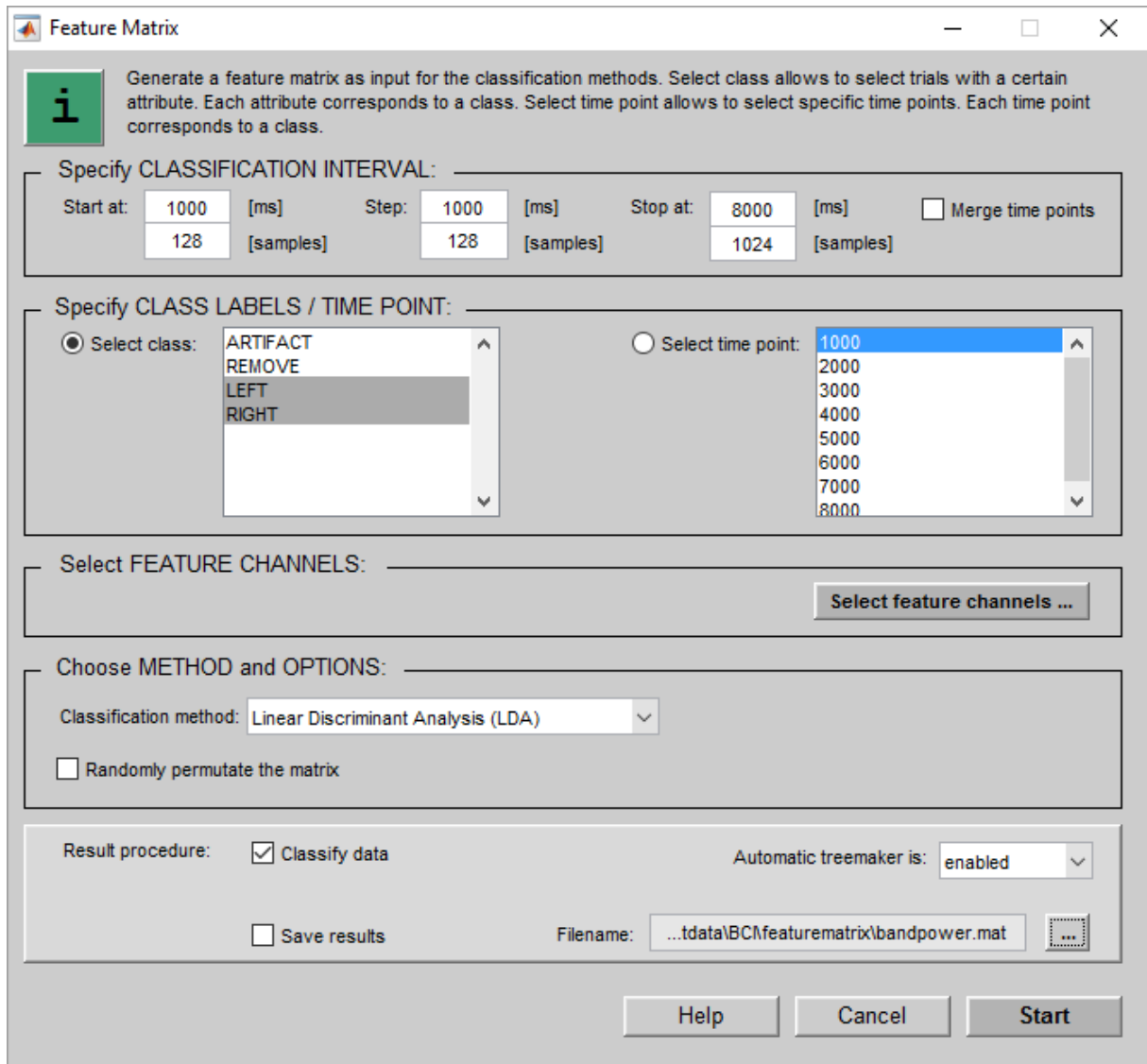
- After finishing the calculation the Data Editor visualizes the newly created features. Channels 4 and 5 represent the bandpower in the alpha range and channels 6 and 7 the bandpower in the beta range.



Trial Attributes

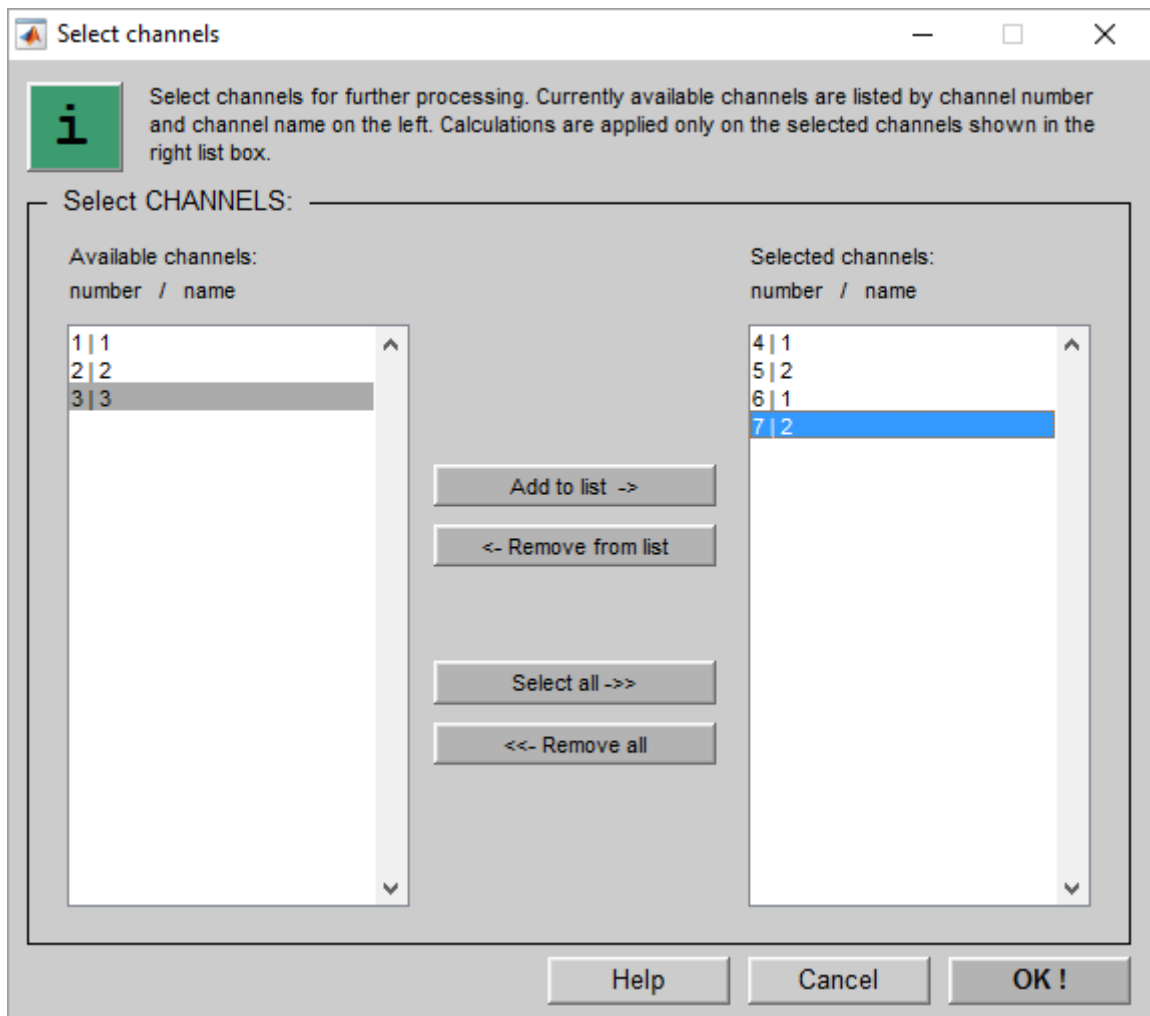
Trial Attributes allows to extract features of trials which correspond to a specific class. The class is assigned with trial attributes in g.BSanalyze (such as right or left hand movement imagination).

1. Open the **Feature Matrix** window from the **Classification** menu



2. The **CLASSIFICATION INTERVAL** should **Start at** 1000 ms with a **Step** size of 1000 ms and **Stop at** 8000 ms. These settings extract for each time point (1000, 2000, ... 8000 ms) the corresponding features.
3. Chose classes `LEFT` and `RIGHT` to extract only trials with these trial attributes

4. Press the **Select features channels ...** button and select the 4 bandpower channels 4, 5, 6 and 7



5. Under **Classification method** it is possible to select a specific method for the classification of the currently generated feature matrix. If the **Classify data** checkbox is enabled the selected classification window will be started.
6. Uncheck the **Randomly permute the matrix** checkbox because the trials with the left and right class labels are already randomly permuted. If this is not the case enable the box to generate a randomly permuted feature matrix.
7. Uncheck the **Classify data** box (if the box is checked the Linear Discriminant Analysis (LDA) window is immediately opened for further processing)
8. Check **Save results** to store the generated feature matrix. The automatic treemaker generates a subdirectory under the current data path with the name `featurematrix`. Enter as filename `bandpower.mat`.
9. Press the **Start** button to generate the feature matrix.

To perform the example demonstrated above from the MATLAB command line use the following code:

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\session1234triggered.mat'];
P_C=load(P_C,File);

% Bandpower
ChannelExclude = [3];
Filter.Name = 'ALPHA';
Filter.Type = 'BP';
Filter.f_low = [7];
Filter.f_high = [13];
Filter.Realization = 'fft';
Filter.Order = [0];
IntervalLength = 128;
Overlap = 127;
Replace = 'add channels';
FileName = '';
ProgressBarFlag = 0;
P_C = gBSbandpower(P_C, ChannelExclude, Filter, IntervalLength,...
    Overlap, Replace, FileName, ProgressBarFlag);

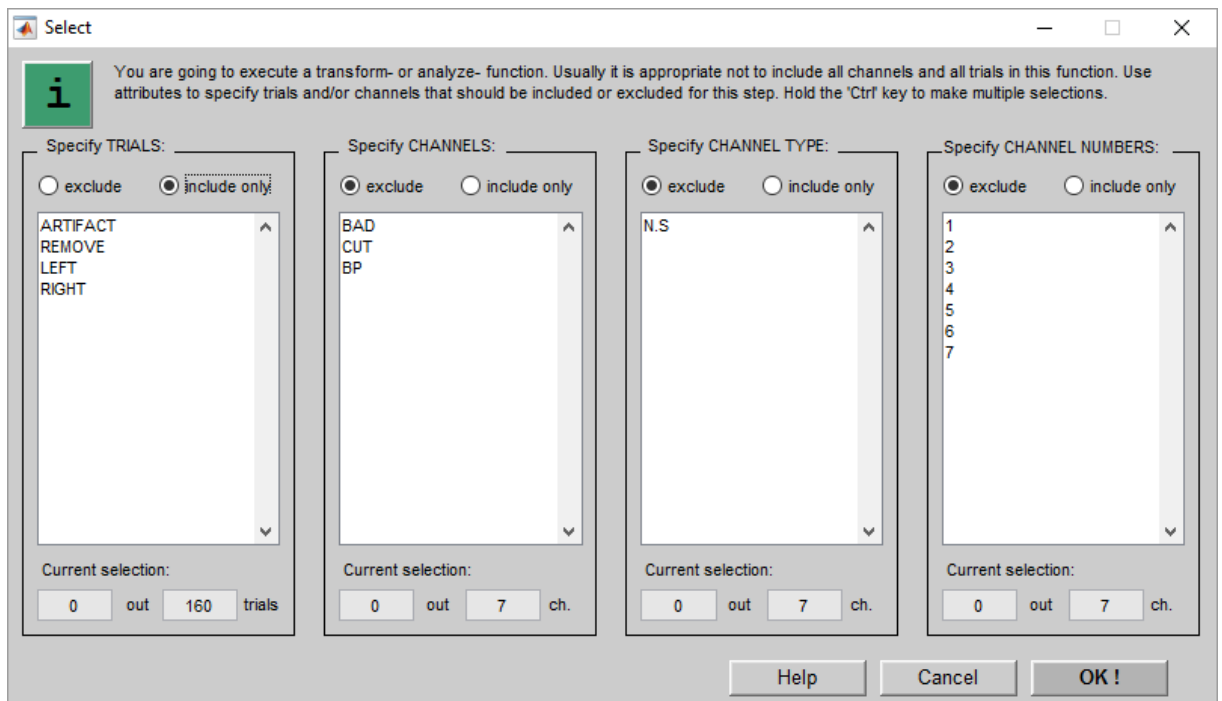
% Bandpower
ChannelExclude = [3 4 5];
Filter.Name = 'BETA-3';
Filter.Type = 'BP';
Filter.f_low = [14];
Filter.f_high = [20];
Filter.Realization = 'fft';
Filter.Order = [0];
IntervalLength = 128;
Overlap = 127;
Replace = 'add channels';
FileName = '';
ProgressBarFlag = 0;
P_C = gBSbandpower(P_C, ChannelExclude, Filter, IntervalLength,...
    Overlap, Replace, FileName, ProgressBarFlag);

%Feature Matrix
Interval=[128 128 1024];
AttributeName={
    'LEFT'
    'RIGHT'
};
ChannelExclude=[1 2 3];
Permutate=0;
MergeTimePoints=0;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\featurematrix\bandpower.mat'];
ProgressBarFlag=[0];
F_O=gBSfeaturematrix(P_C,Interval,AttributeName,Permutate,...
MergeTimePoints,ChannelExclude,FileName,ProgressBarFlag);
```

Time Points

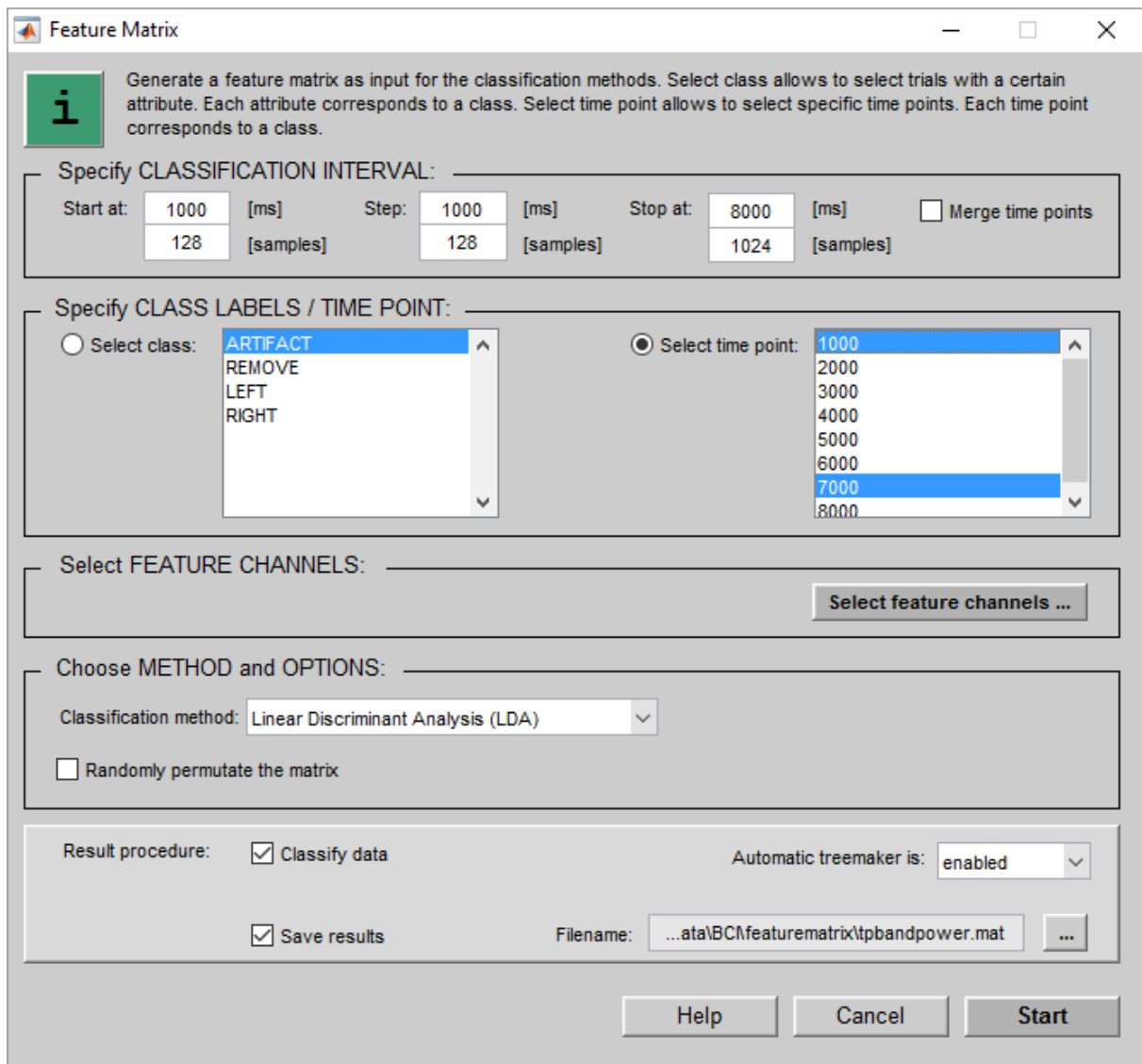
To extract features of the signal at different time points to investigate e.g. an initial state and an active state perform the following steps.

1. Load the data-set `session1234bp.mat` that was created in the previous section from
`Documents\gtec\gBSanalyze\testdata\BCI\session1234bp.mat`
2. Open **Cut Trials Channels** from the **Transform** menu and click the **Select trials/chan.** button. In **Specify TRIALS** this dialog click the **Include only** radio button and select only `RIGHT` trials for further operation. This allows inspecting changes between the inactive and active state of the BCI experiment for all right hand movement imagination trials.



3. After finishing the settings click **OK** to close the **Select** dialog.
4. Click **Start** to perform the action with the settings provided above.

5. Open the **Feature Matrix** window from the **Classification** menu



6. Set the **CLASSIFICATION INTERVAL Start at** box to 1000 ms, the **Step** box to 1000 ms and the **Stop at** box to 8000 ms. These settings populate the **Select time point** listbox with specific time points.
7. Select 2000 ms and 7000 ms to extract only the features at these time points
8. Check the **Save results** box to store the features under `tpbandpower.mat`. The automatic treemaker generates the path.
9. Press **Start** to perform the operation.
10. The **Linear Classifier** dialog will open, see corresponding chapter for further description.

To perform the example demonstrated above from the MATLAB command line use the following code:

```
%Load Data
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\session1234bp.mat'];
P_C=load(P_C,File);

%Select Trials and Channels
trial_id=[4];
channel_id=[];
type_id=[];
channelnr_id=[];
flag_tr='tr_inc';
flag_ch='ch_exc';
flag_type='type_exc';
flag_nr='nr_exc';
[TrialExclude, ChannelExclude]=gBSselect(P_C,trial_id,...
flag_tr,channel_id,flag_ch,type_id,flag_type,channelnr_id,flag_nr);

P_C=gBScuttrialschannels(P_C,TrialExclude,ChannelExclude);

%Feature Matrix
Interval=[256 896];
AttributeName={};
ChannelExclude=[];
Permutate=0;
MergeTimePoints=0;
FileName= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\featurematrix\tpbandpower.mat'];
ProgressBarFlag=[0];
F_O=gBSfeaturematrix(P_C,Interval,AttributeName,Permutate,...
MergeTimePoints,ChannelExclude,FileName,ProgressBarFlag);
```

Time Segments

Time Segments allows to extract multiple segments of a trial to a feature matrix. Each segment of a trial receives its own class label. Therefore, if three segments are extracted the feature matrix contains three classes.

Follow these steps to generate a time segment feature matrix:

1. Load the data file `Data.mat` from

`Documents\gtec\gBSanalyze\testdata\SelfPaced`

into the Data Editor. The Data Editor shows one ECoG channel and one trigger impulse channel.

3. Check **Set-start marker** to assign an OR1 marker to each rising edge on the trigger channel. Uncheck the **Show epoching areas** box.
4. To search only on the trigger impulse channel press the **Select trials /chan.** button and select channel 2
5. Press **Start** to search for the trigger impulses.

The Data Editor shows now 50 markers.

6. Open the **Trigger** window from the **Transform** menu and check the **Marker** radio button and select OR1 to extract 6 second trials around each OR1 marker

Trigger

The trigger function splits your recorded data into trials related to trigger timepoints defined by physical channels or markers. The use of different markers or channels allows you to assign attributes automatically to resulting trials. Channel attributes and markers remain in the triggered data.

Specify TRIAL PARAMETERS:

Time before trigger: 2000 [ms] / 400 [samples] Time after trigger: 4000 [ms] / 800 [samples] Accept incomplete last trial

Specify TRIGGERS and ATTRIBUTES:

Physical channel: 1(CH1) / 2(CH2) Edge: rising Threshold voltage: 201.1 [µV]
 Marker: OR1 Threshold level: 90 [% of max.]
 Slew rate: 24.67 [µV/ms]

Assign attribute to resulting trials:

Accept overlap add to list -> <- remove from list apply changes ->

Change color: red blue green yellow pink orange purple olive brown grey

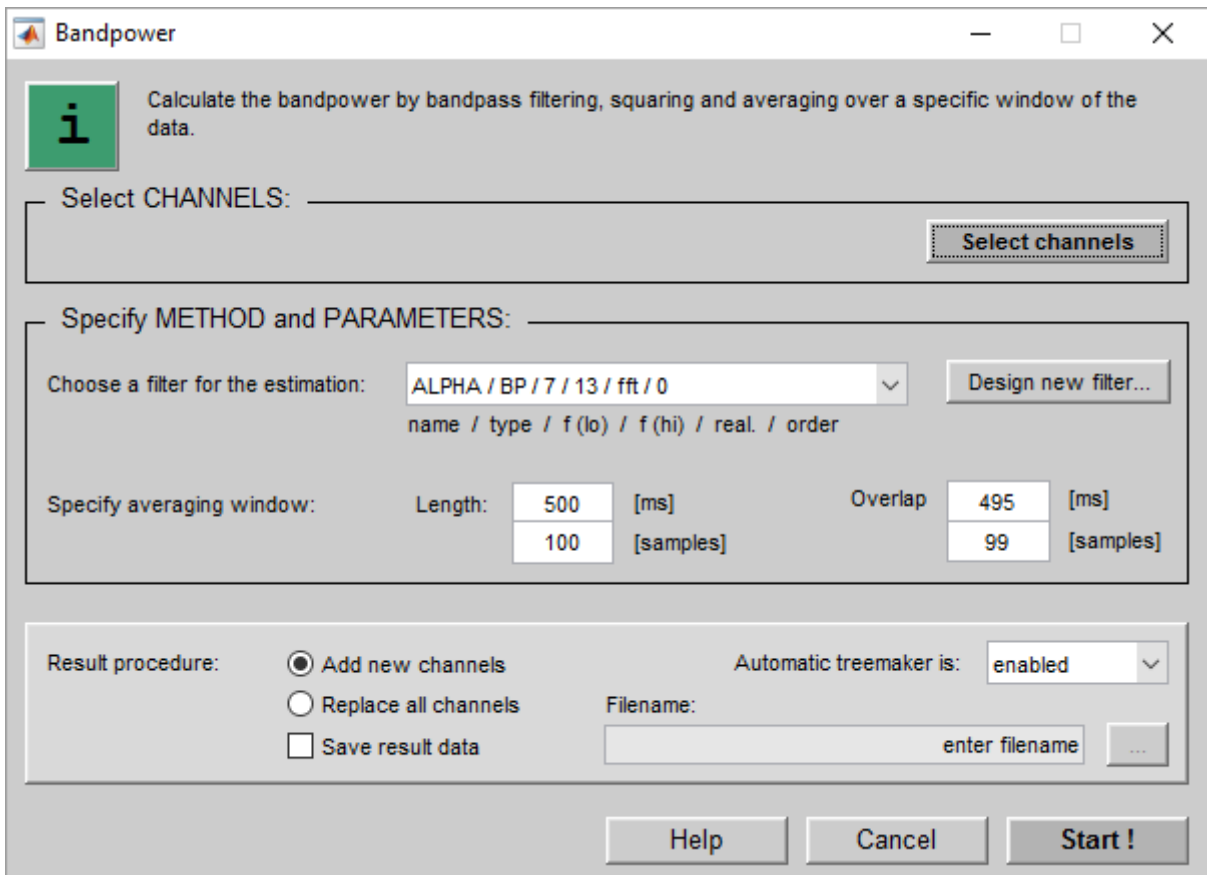
Chan. (Marker)	Name	Edge	Value	Attribute	Overlap	Color
mark/ OR1	- / - / - /	- / - /	yes	red		

Generate LINED UP TRIALS:

Line up trials (no trigger) Length of trials: 1000 [ms] / 200 [samples] Overlap: 0 [ms] / 0 [samples]

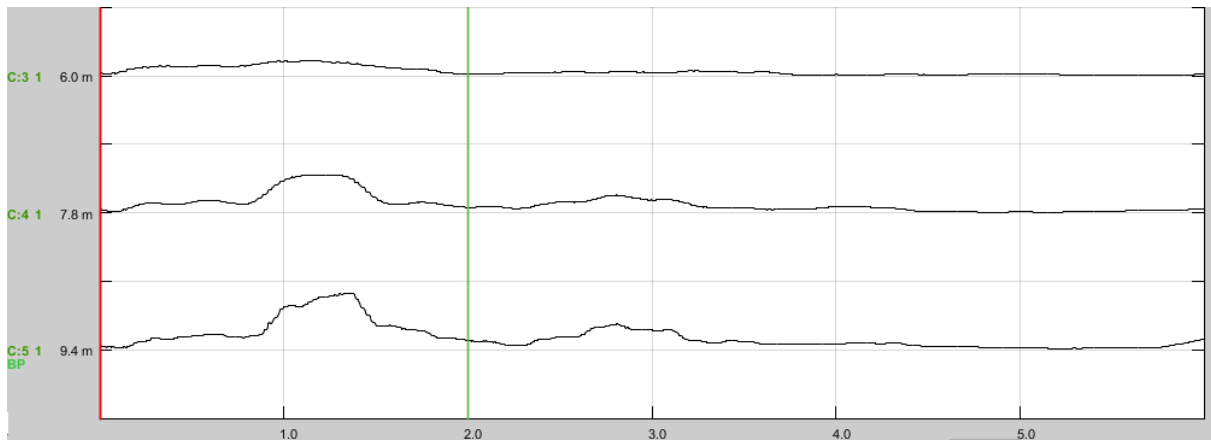
Select CHANNELS for the triggered file:

7. Check the **Accept Overlap** box to allow overlapping trials
8. Press the **add to list** button to accept this trigger criterion
9. Press **Start !** to perform the operation. Now the Data Editor shows 49 trials with a trial length of 6 seconds. 2 seconds prior to the trigger impulse and 4 seconds after the impulse.
10. To calculate parameters from the ECoG channel open the **Bandpower** window from the **Parameter Extraction** menu



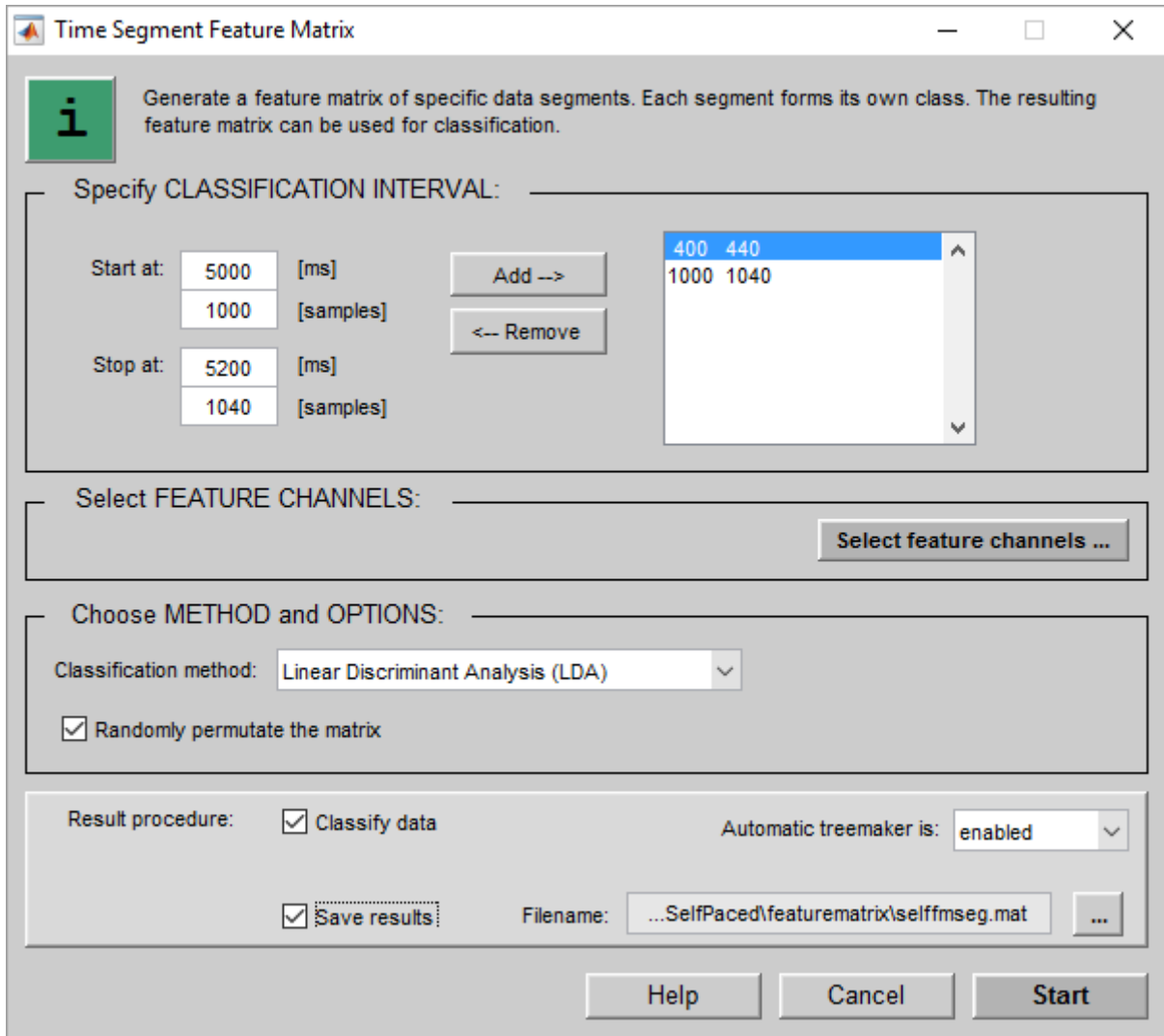
11. Select the ALPHA filter to extract the bandpower from 7 to 13 Hz
12. Press the **Select channels** button and chose only channel 1 for the calculation
13. Select **Add new channels** to append the new feature channel to the data in the Data Editor
14. Press **Start** to perform the calculation
15. Repeat steps 10 to 14 with the BETA-3 and with the BETA filter

The Data Editor contain now 3 additional bandpower channels



16. Open the **Time Segment Feature Matrix** window from the **Classification** menu and define the classification segments. Enter under **Start at** 2000 ms and under **Stop at** 2200 ms and press the **Add** button. Then enter 5000 ms and 5200 ms and press again the **Add** button. These settings will extract two segments from each trial. The first segment will be the first class and the second segment the second class.
17. Press the **Select features channels ...** button and chose the bandpower channels 3, 4 and 5
18. Check the **Randomly permutate the matrix** box to generate a random permutation of the trials

19. Click on **Save results** and enter the name `selffmseg.mat` into the upcoming window to store the feature matrix.



20. If the **Classify data** box is checked the **Linear Discriminant Analysis (LDA)** algorithm window would be opened for further processing, see chapter **Linear Classifier** for further description.

21. Press **Start** to perform the operation.

To perform the example demonstrated above from the MATLAB command line use the following code:

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\SelfPaced\Data.mat'];
P_C=load(P_C,File);

%Select Trials and Channels
trial_id=[];
channel_id=[];
type_id=[];
channelnr_id=[2];
flag_tr='tr_exc';
flag_ch='ch_exc';
flag_type='type_exc';
flag_nr='nr_inc';
[TrialExclude, ChannelExclude]=gBSselect(P_C,trial_id,flag_tr,...
channel_id,flag_ch,type_id,flag_type,channelnr_id,flag_nr);

% Eventfinder (overflow)
MarkOverflow = 1;
showEpochingAreas_over = 0;
setStartMarker_over = 1;
setStopMarker_over = 0;
AssignAttribute_over = 0;
StartMarker_over = 'OR1';
StopMarker_over = 'OR2';
TrialAttribute_over = 'OVERRUN';
Threshold_over = 90;
getUnit_over = '% of max';
TrialExclude_over = [];
ChannelExclude_over = [1];
ProgressBarFlag = 0;
[P_C, PreviewOverflow, VecThreshold] = gBSoverflow...
(P_C, MarkOverflow, showEpochingAreas_over,...
setStartMarker_over, setStopMarker_over,...
AssignAttribute_over, StartMarker_over, StopMarker_over,...
TrialAttribute_over, Threshold_over, getUnit_over,...
TrialExclude_over, ChannelExclude_over, ProgressBarFlag);

%Trigger
New_tm{1}={3 1};
SamplesBefore=400;
SamplesAfter=800;
Uncomplete=0;
ChannelExclude=[];
P_C=gBStrigger(P_C,New_tm,SamplesBefore,SamplesAfter,Uncomplete,ChannelExcl
ude);
```



```

% Bandpower
ChannelExclude = [2];
Filter.Name = 'ALPHA';
Filter.Type = 'BP';
Filter.f_low = [7];
Filter.f_high = [13];
Filter.Realization = 'fft';
Filter.Order = [0];
IntervalLength = 100;
Overlap = 99;
Replace = 'add channels';
FileName = '';
ProgressBarFlag = 0;
P_C = gBSbandpower(P_C, ChannelExclude, Filter, IntervalLength,...
                  Overlap, Replace, FileName, ProgressBarFlag);

% Bandpower
ChannelExclude = [2 3];
Filter.Name = 'BETA-3';
Filter.Type = 'BP';
Filter.f_low = [14];
Filter.f_high = [20];
Filter.Realization = 'fft';
Filter.Order = [0];
IntervalLength = 100;
Overlap = 99;
Replace = 'add channels';
FileName = '';
ProgressBarFlag = 0;
P_C = gBSbandpower(P_C, ChannelExclude, Filter, IntervalLength,...
                  Overlap, Replace, FileName, ProgressBarFlag);

% Bandpower
ChannelExclude = [2 3 4];
Filter.Name = 'BETA';
Filter.Type = 'BP';
Filter.f_low = [14];
Filter.f_high = [32];
Filter.Realization = 'fft';
Filter.Order = [0];
IntervalLength = 100;
Overlap = 99;
Replace = 'add channels';
FileName = '';
ProgressBarFlag = 0;
P_C = gBSbandpower(P_C, ChannelExclude, Filter, IntervalLength,...
                  Overlap, Replace, FileName, ProgressBarFlag);

%Time Segment Feature Matrix
Interval=[
400 440
1000 1040
];
ChannelExclude=[1 2];
Permutate=1;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\SelfPaced\featurematrix\selffmseg.mat'
];
ProgressBarFlag=[0];
F_O=gBStimesegmentfeaturematrix(P_C,Interval,Permutate,...
ChannelExclude,FileName,ProgressBarFlag);

```

Generating a Classifier

This section explains the classification and classifier generation of the feature matrix with the following methods:

Linear Classifier

- Multi-Class Linear Discriminant Analysis (LDA)
- Minimum Distance Classifier (MDC)

Neural Network

- Multi-Layer Perceptron (MLP)
- Radial Basis Function (RBF)

*DSL*VQ

- Distinction Sensitive Learning Vector Quantization (DSL

All classification windows have certain control fields in common:

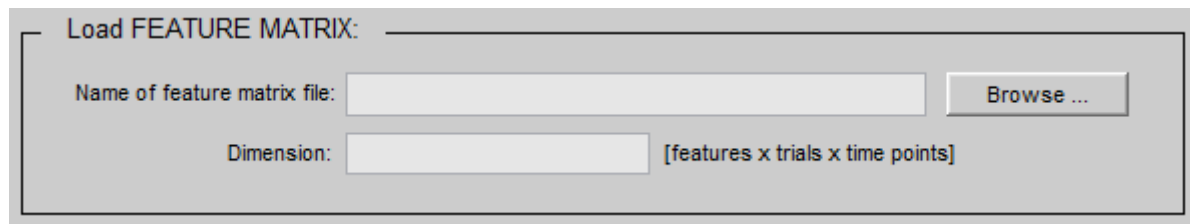
The **Load FEATURE MATRIX** field allows to **Browse** for a feature matrix file and shows the filename as well as the **Dimension** of the matrix. The number of features corresponds to the number of channels used for the feature matrix generation.

The number of trials depends on the feature matrix generation mode:

Mode 1 – Trial Attribute ... the number corresponds to the number of trials which have a certain trial attribute (e.g. left or right)

Mode 2 – Time Point ... the number corresponds to the number of selected time points (e.g. 2000 ms and 7000 ms) times the number of trials (e.g. 80)

Mode 3 – Time Segment ... the number corresponds to the number of samples in the time segments times the number of trials



Load FEATURE MATRIX:

Name of feature matrix file:

Dimension: [features x trials x time points]

Time points expresses the generation time points of the feature matrix. For each single time point the whole feature matrix is generated (e.g. 1000, 2000, ... 8000 ms).

The **Select FEATURE CHANNELS** field allows to specify the numbers (corresponds to the channel number when the feature matrix was generated) of the features that should be used for plotting a cloud of the feature matrix in gResult2d.

Choose METHOD and OPTIONS allows to select the classification method and the training and test-sets:

10 x 10 cross validation ... The 10 times 10 fold cross validation mixes the data set randomly and divides it into 10 equally sized distinct partitions. Each partition is then used

once for testing, the other partitions are used for training. This results in 10 different error rates, which are averaged. This is the error rate of a 10 fold cross validation. To further improve the estimate the procedure is repeated 10 times and again all error rates are averaged.

Train 50 % - Test 50 % ... uses the first 50 % of the feature matrix for training and the rest for testing

Train 100 % - Test 100 % ... uses all the data for training and testing

Train 100 % - Test 0 % ... uses all the data for training. This is useful to generate a classifier.

The **Result procedure** field allows to open gResult2d with the classification result and a cloud of the features. The **classifier window** option opens the MATLAB Editor with classification results and classifiers (weight vectors). **Save results** allows to store the classification result under a specific filename. If the **Automatic treemaker** is enabled a subdirectory under the current data directory is created. The result is stored into this subdirectory.

Linear Classifier

The Linear Classifier window allows to perform a linear discriminant analysis and minimum distance classifier analysis of multiple classes.

Multi-Class LDA

Linear Discriminant of Fisher

An optimal decision rule for minimizing the probability of misclassification is based on the idea of discriminant functions. The simplest form consists of a linear combination of the inputs. The parameters are obtained with a learning algorithm from a set of training data. Fisher introduced a method that reduces the dimensionality before classification [Bishop 1995].

The dimension reduction is done by projecting the input data \mathbf{x} onto a value y with adjustable weights \mathbf{w}

$$y = \mathbf{w}^T \mathbf{x} \quad (1)$$

Of course this leads to a loss of information but we chose \mathbf{w} in such a way that maximizes the class separation between class1 and class 2 (e.g. left and right finger movement).

For class „left finger“ the mean vector is

$$\mathbf{m1} = \frac{1}{N1} \sum_{n \in \text{Classleftfinger}} \mathbf{x}_{n\text{leftfinger}} \quad (2)$$

where $N1$ is the length of the input vector.

For the class „right hand“ $\mathbf{m2}$ is

$$\mathbf{m2} = \frac{1}{N2} \sum_{n \in \text{Classrightfinger}} \mathbf{x}_{n\text{rightfinger}} \quad (3)$$

The separation of the two classes is made by separating $\mathbf{m1}$ and $\mathbf{m2}$

$$\mathbf{m2} - \mathbf{m1} = \mathbf{w}^T (\mathbf{m2} - \mathbf{m1}) \quad (4)$$

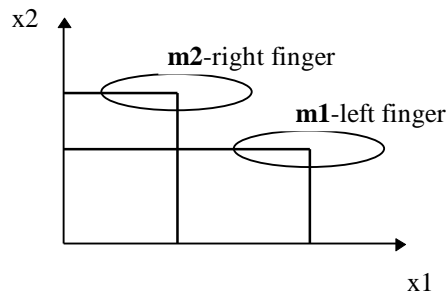
By choosing \mathbf{w} arbitrary large the difference increases, therefore define

$$\sum_i \mathbf{w}_i^2 = 1 \quad (5)$$

and after some calculation we obtain

$$w \propto (\mathbf{m2} - \mathbf{m1}) \quad (6)$$

But a problem arises with this separation that is shown below.



Separation problem.

If we project $\mathbf{m1}$ and $\mathbf{m2}$ onto the $x1$ axes the difference is bigger than in the case of projecting onto the $x2$ axes. But there are within-class spreads that cause a better separation when $\mathbf{m1}$ and $\mathbf{m2}$ are projected onto $x2$.

Fisher proposed as solution

$$\mathbf{w} \propto \mathbf{S}_w^{-1}(\mathbf{m2} - \mathbf{m1}) \quad (7)$$

where \mathbf{S}_w is the total within class covariance matrix

$$\mathbf{S}_w = \sum_{n \in \text{Classleftfinger}} (\mathbf{x}_n - \mathbf{m1})(\mathbf{x}_n - \mathbf{m1})^T + \sum_{n \in \text{Classrightfinger}} (\mathbf{x}_n - \mathbf{m2})(\mathbf{x}_n - \mathbf{m2})^T \quad (8)$$

Which is a projection rule for the data down to one dimension.

By choosing a threshold y_0 we can classify a point to class 1 if it is greater zero or to class 2 otherwise.

Obviously the dimension reduction reduces the amount of information, but it can lead to improvements of the classifier performance [Bishop 1995].

References:

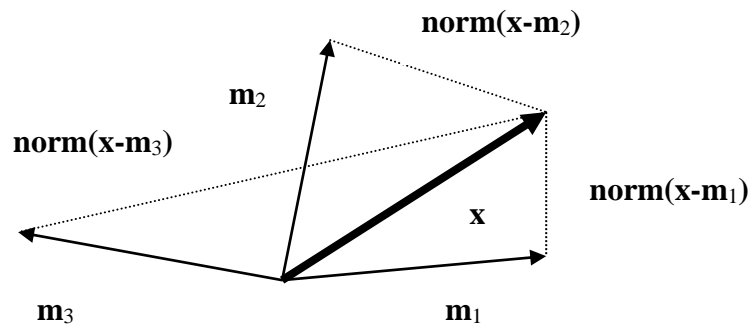
Bishop, C. M., Neural Networks for Pattern Recognition. Clarendon Press, Oxford, 1995.

Minimum Distance Classifier

Let \mathbf{x} be the feature vector for the unknown input. Vectors $\mathbf{m}_1, \mathbf{m}_2, \dots, \mathbf{m}_c$ are the templates (i.e., perfect, noise-free feature vectors) for the c classes. Then the error in matching \mathbf{x} against \mathbf{m}_k is given by

$$\|\mathbf{x} - \mathbf{m}_k\|.$$

where $\|\mathbf{u}\|$ is called the **norm** of the vector \mathbf{u} . The minimum-error classifier calculates $\|\mathbf{x} - \mathbf{m}_k\|$ for $k = 1$ to c to find the class for which this error is minimum. $\|\mathbf{x} - \mathbf{m}_k\|$ is also the distance from \mathbf{x} to \mathbf{m}_k and therefore the method is called **minimum-distance** classifier.



Linear Classifier window:

Classification method can be Linear Discriminant Analysis (LDA) or Minimum Distance Classifier (MDC). In the case of MDC under **Metric** the Mahalanobis or Euclidian distance can be selected.

Linear Classifier

Compute a classifier (weight vector) between groups of trials marked by class labels (trial attributes). The performance of the classifier can be estimated by cross-validation. The weight vector can be viewed, edited and stored for real-time processing with q.RTsys.

Load FEATURE MATRIX:

Name of feature matrix file:

Dimension: [features x trials x time points]

Select FEATURE CHANNELS:

Map feature no.: against:

Choose METHOD and OPTIONS:

Classification method: Metric: Mahalanobis distance
 Euclidian distance

Training / test-sets:

Result procedure: Show with Result2D Open classifier window Save results

Automatic treemaker is:

Filename:

Example 1:

Perform the following steps to make a classification of an EEG-based brain-computer interface data-set:

1. Open the window **Linear Classifier** from the **Classification** menu
2. Press the **Browse** button and select the feature matrix file `bandpower.mat` that was created in the previous example and is stored under

`Documents\gtec\gBSanalyze\testdata\BCI\featurematrix`

The featurematrix contains 4 features (2 bandpower values in the alpha range and 2 bandpower values in the beta range), 160 trials (80 right and 80 left) and 8 time points (1000, 2000, ... 8000 ms).

3. Under **Select FEATURE CHANNELS** select **Map feature no 3 against 4** to plot a cloud of the feature matrix in `gResult2d`
4. Select `Linear Discriminant Analysis (LDA)` from the pull-down menu

Linear Classifier

Compute a classifier (weight vector) between groups of trials marked by class labels (trial attributes). The performance of the classifier can be estimated by cross-validation. The weight vector can be viewed, edited and stored for real-time processing with q.RTsys.

Load FEATURE MATRIX:

Name of feature matrix file: `...alyze\testdata\BCI\featurematrix\bandpower.mat`

Dimension: `4 160 8` [features x trials x time points]

Select FEATURE CHANNELS:

Map feature no.: `3` against: `4`

Choose METHOD and OPTIONS:

Classification method: `Linear Discriminant Analysis (LDA)` Metric: Mahalanobis distance

Training / test-sets: `10 x 10 cross-validation` Euclidian distance

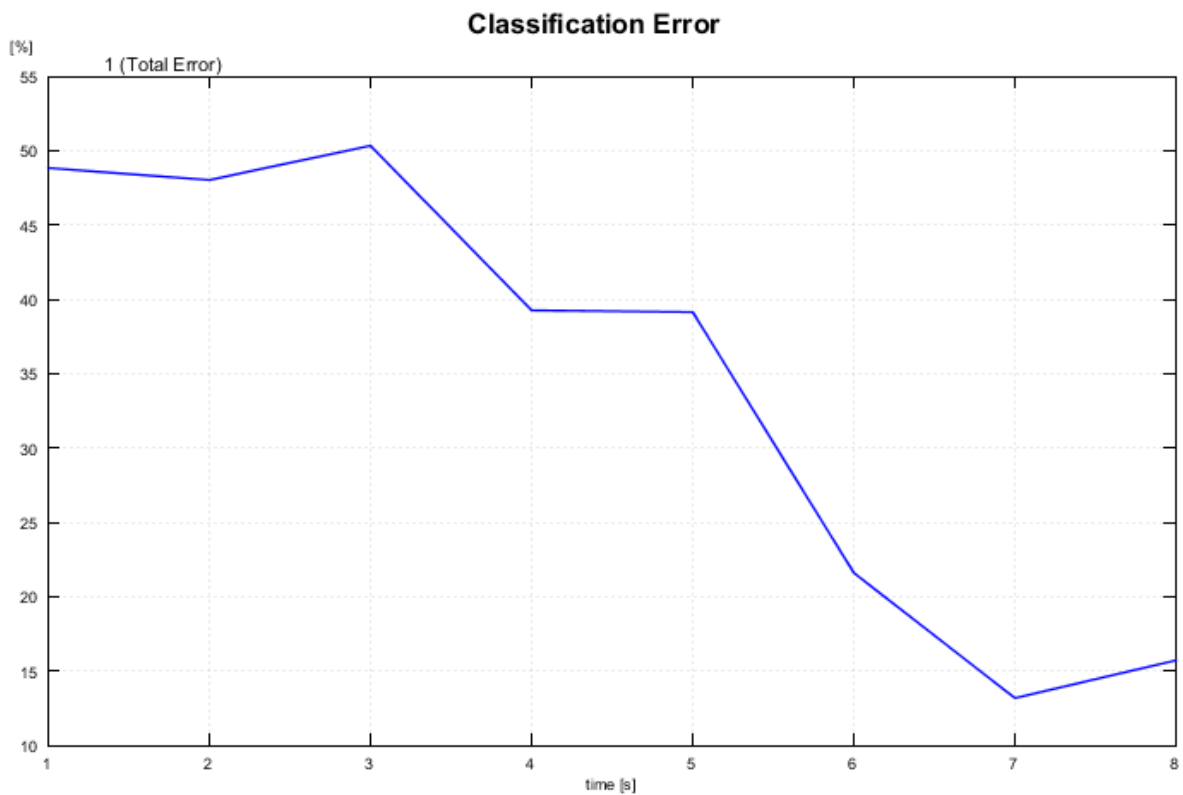
Result procedure: Show with Result2D Automatic treemaker is: `enabled`

Open classifier window

Save results Filename: `...stdata\BCI\featurematrix\lc\LDAbp.mat`

5. Select **10 x 10 cross-validation** to randomly mix the training and testing data-set
6. Check the **Show with Result2D** and **Open classifier window** to open gResult2d and the MATLAB Editor to view the classification result and the weight vector (not available for 10 x 10 cross validation)
7. To store the classification result check **Save results** and enter the filename LDAbp.mat. The automatic treemaker generates the directory.

gResult2d opens with the classification result. The classification error is at the beginning around 50 % and drops down to 13 % at second 7.



The MATLAB Editor shows the ASCII description of the classification. The first and second columns of the matrix shows the classification time point in seconds and samples. The third column represents the mean classification error followed by the standard deviation. The following columns give the classification errors of the 10 cross-validation runs.

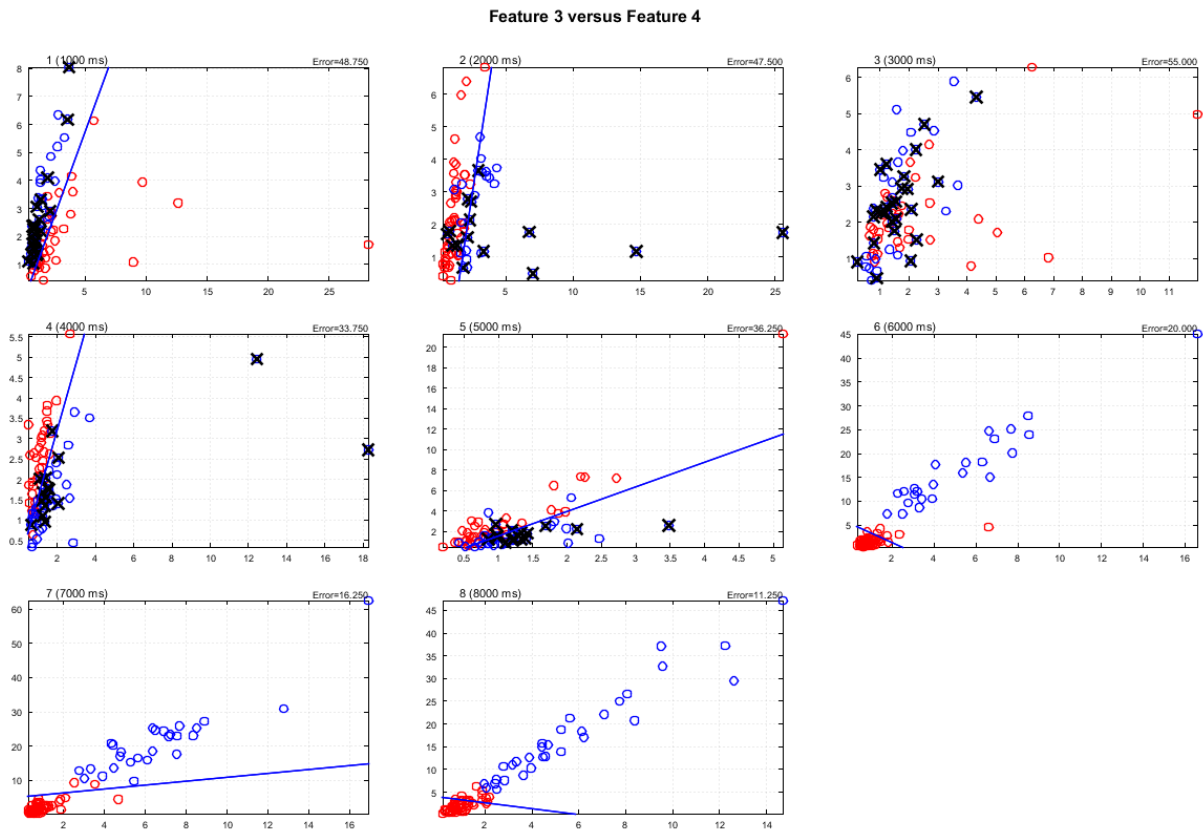
```
Classification Method: Linear Discriminant Analysis
Training- and testdata option: CV
class 1: LEFT
class 2: RIGHT
Total Nr. of Trials: 160
Trials per selected Class 1: 80
Trials per selected Class 2: 80
```

Second	Sample	Mean Error	Std	[Total Error	CV	Runs]								
1.000	128.000	48.3	2.6	48.8	46.3	46.9	50.0	50.6	50.0	52.5	44.4	45.0	48.8	
2.000	256.000	46.8	1.4	44.4	46.9	46.3	48.1	46.3	47.5	48.8	46.3	45.0	48.1	
3.000	384.000	49.6	1.8	50.6	50.0	49.4	52.5	48.1	50.6	48.8	47.5	51.2	46.9	
4.000	512.000	39.1	1.2	39.4	41.3	39.4	39.4	38.8	37.5	38.1	39.4	40.0	37.5	
5.000	640.000	39.4	1.1	40.6	39.4	40.0	40.6	39.4	39.4	36.9	38.8	39.4	39.4	
6.000	768.000	21.3	0.6	21.9	21.3	21.9	20.6	21.3	20.6	21.3	20.6	21.3	22.5	
7.000	896.000	13.0	0.4	12.5	12.5	13.1	12.5	13.1	13.1	13.1	13.1	13.1	13.8	13.1
8.000	1024.000	15.3	0.4	15.0	15.0	15.0	15.6	15.6	15.6	14.4	15.0	15.6	15.6	

8. Repeat steps 1 to 7 but select under **Training / test-sets** Train 50 % - Test 50 %

gResult2d opens with the classification result on page 1. Change to page 2 of gResult2d to view the cloud of feature 3 versus feature 4. The red circles show the feature of a LEFT trial, the blue circles of a RIGHT trial. The blue line represents the weight vector of the linear discriminant analysis. The black crosses show the wrong classified trials.

At second 1 the classification error is 48,75 % and therefore the left and right classes can not be differentiated. But at second 8 the discrimination is possible with an error of 11.25 %. Note that the blue and red circles are clearly separated.



Now the MATLAB Editor shows also the weight vectors for each calculated time point. Under Classifier 1 the bias value of the LDA classifier can be found. Classifier 2 shows the weight values for each feature channel in the same sequence as the features were extracted in the Data Editor.

```
Second/Sample: 8.000 1024.000
Classifier 1
0.565 |
Classifier 2
0.336 |
-0.098 |
-0.090 |
-0.139 |
```

Furthermore, the trial number of the wrong classified trials is given for each time point. In this case only trials of class 2 were wrong classified.

```
Second/Sample: 8.000 1024.000
Class Number/Nr Trials:
Class 1:
Class 2: 83 86 91 92 97 101 123 124 155
```

To perform the example demonstrated above from the MATLAB command line use the following code:

```
%Load FeatureMatrix
F_M=featurematrix;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\featurematrix\bandpower.mat'];
F_M=load(F_M,FileName);

%Linear Classifier
PlotFeatures=[1 2];
Method=['LDA'];
P.metric=[''];
TrainTestData=['CV'];
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\featurematrix\lc\LDAbp.mat'];
ProgressBarFlag=[0];
C_O=gBSlinearclassifier(F_M,Method,P,TrainTestData,PlotFeatures,...
FileName,ProgressBarFlag);

%Linear Classifier
PlotFeatures=[3 4];
Method=['LDA'];
P.metric=[''];
TrainTestData=['50:50'];
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\featurematrix\lc\LDAbp.mat'];
ProgressBarFlag=[0];
C_O=gBSlinearclassifier(F_M,Method,P,TrainTestData,PlotFeatures,...
FileName,ProgressBarFlag);
```

Example 2:

This example shows the classification of a three class problem with the Minimum Distance Classifier (MDC).

1. Load the data file `3classes.mat` into the Data Editor from

`Documents\gtec\gBSanalyze\testdata\Classify`

The data-set contains 150 trials with 2 feature channels. The channels contain artificial generated random numbers and the classes are separated by specific mean values. Trial 128 represents an outlier.

2. Open **Feature Matrix** from the classification menu and set the **CLASSIFICATION INTERVAL** to **Start at** 1000 ms, **Step** 1000 ms and **Stop at** 5000 ms.

Feature Matrix

Generate a feature matrix as input for the classification methods. Select class allows to select trials with a certain attribute. Each attribute corresponds to a class. Select time point allows to select specific time points. Each time point corresponds to a class.

Specify CLASSIFICATION INTERVAL:

Start at: 1000 [ms] / 128 [samples] Step: 1000 [ms] / 128 [samples] Stop at: 5000 [ms] / 640 [samples] Merge time points

Specify CLASS LABELS / TIME POINT:

Select class: ARTIFACT, REMOVE, 3, 4, 5

Select time point: 1000, 2000, 3000, 4000, 5000

Select FEATURE CHANNELS:

Choose METHOD and OPTIONS:

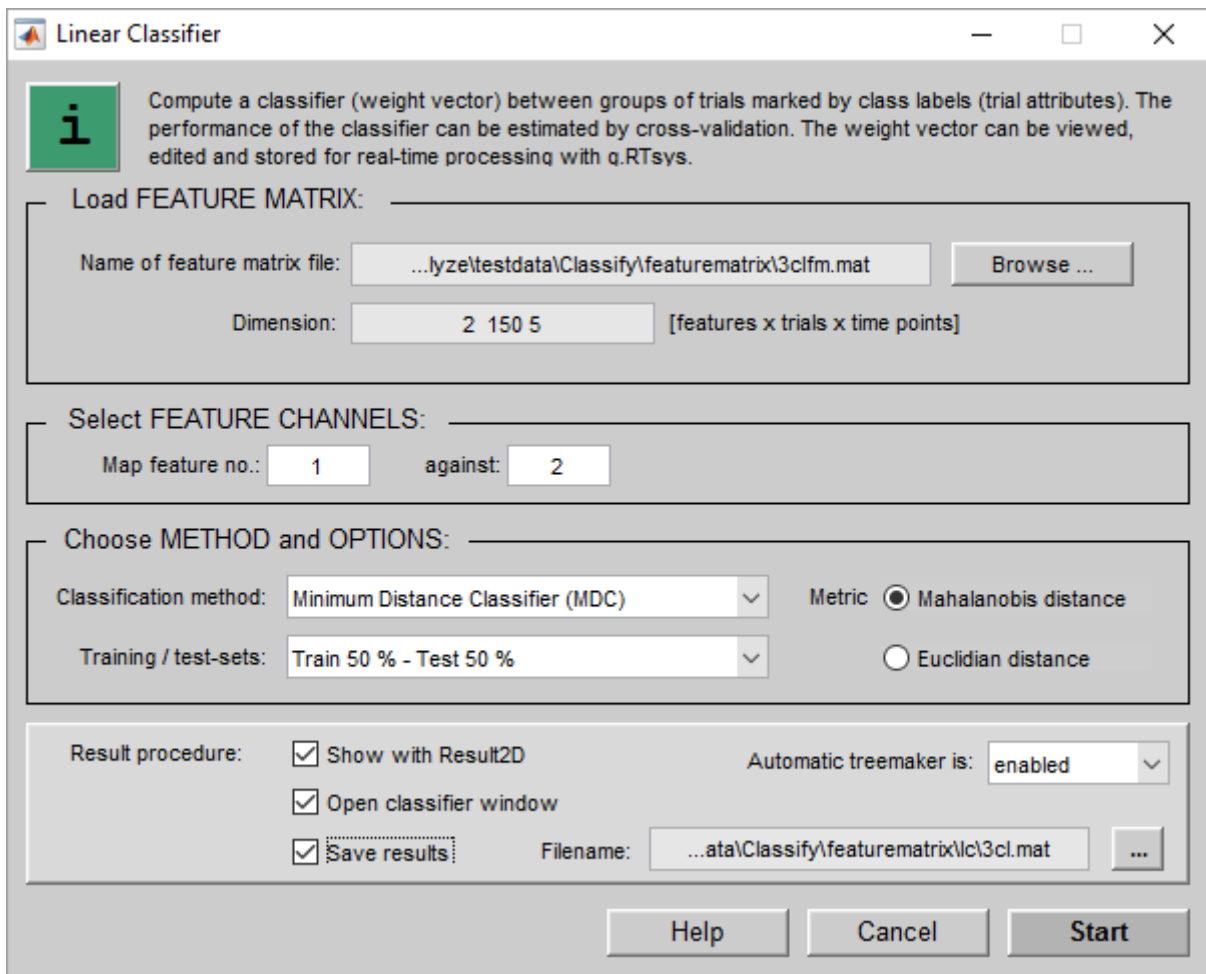
Classification method: Minimum Distance Classifier (MDC) Randomly permutate the matrix

Result procedure: Classify data Automatic treemaker is: enabled

Save results Filename: ...data\Classify\featurematrix\3clfm.mat

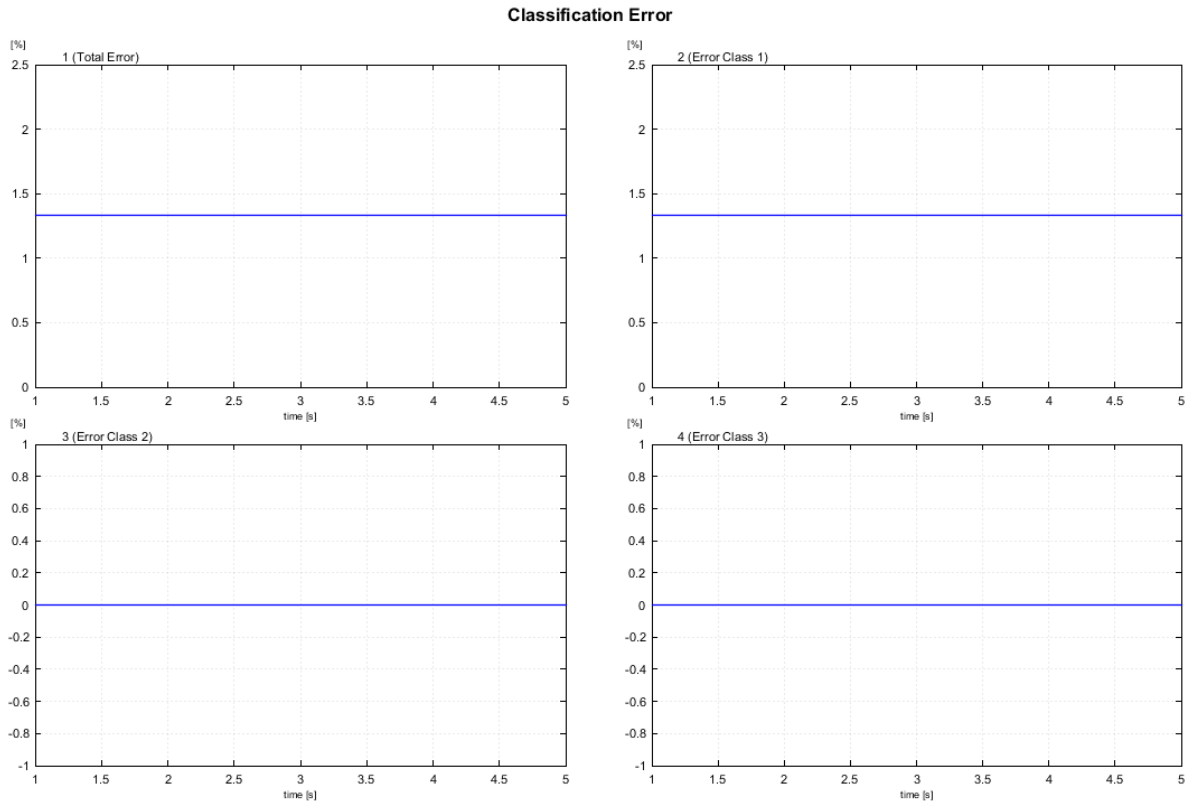
3. Select classes 3, 4 and 5 to separate all 3 classes
4. Chose **Minimum Distance Classifier (MDC)** under **Classification method**
5. Check **Save results** and enter `3clfm.mat` as filename
6. Press **Start** to extract the feature matrix and to open the **Linear Classifier** window

The dimension of the feature matrix is 2 features (2 channels), 150 trials and 5 time points (1000, 2000,...5000ms).

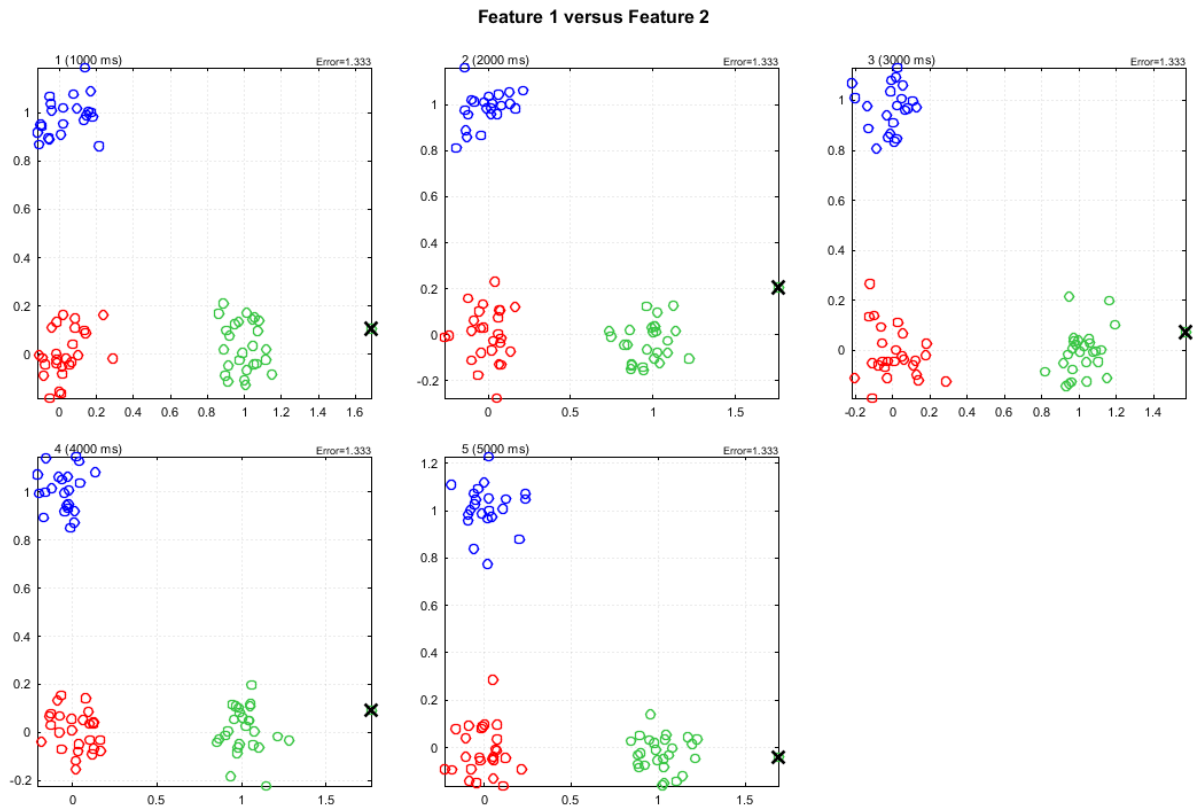


7. Select **Minimum Distance Classifier (MDC)** under **Classification method** and chose **Train 50 % - Test 50 %**
8. Press **Start** to perform the classification

Page 1 of gResult2d shows 4 classification error time courses. The first channel shows the **Total Error** of all classes. Channels 2 to 4 represent the error rates for each individual class. In this case only class 1 has an error rate which is not zero. This is the case because trial 128 is marked as class 1 trial but represents an outlier. Note that only 50 % of the trials are used for testing and therefore the total trial number is 75 trials.



The second page maps feature 1 versus feature 2. A color is assigned to each class and the wrong classified trial 128 is indicated by the black cross. Trial 128 was classified as green class but belongs to the red class. Therefore, the linear method is not able to correctly identify trial 128.



The MATLAB Editor shows that 27 trials were selected of class 1, 22 of class 2 and 26 of class 3.

```

Classification Method: Linear Discriminant Analysis
Training- and testdata option: 50:50
class 1: 3
class 2: 4
class 3: 5
Total Nr. of Trials: 75
Trials per selected Class 1: 27
Trials per selected Class 2: 22
Trials per selected Class 3: 26

```

The weight vector is given for all three classes. The symbol “|” is used to separate the classifiers.

```

Second/Sample: 5.000 640.000
Classifier 1
 1.000 0.000 | 1.000 0.000 | 1.000 0.000 |
 0.000 1.000 | 0.000 1.000 | 0.000 1.000 |
Classifier 2
 0.032 | 0.032 | 1.016 |
-0.010 | 0.997 | 0.006 |

```

To perform the example demonstrated above from the MATLAB command line use the following code:

%Load Data

```

P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Classify\3classes.mat'];
P_C=load(P_C,File);

```

%Feature Matrix

```

Interval=[128 128 640];
AttributeName={
    '3'
    '4'
    '5'
};
ChannelExclude=[];
Permutate=0;
MergeTimePoints=0;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Classify\featurematrix\3clfm.mat'];
ProgressBarFlag=[0];
F_O=gBSfeaturematrix(P_C,Interval,AttributeName,Permutate,...
MergeTimePoints,ChannelExclude,FileName,ProgressBarFlag);

```

%Load FeatureMatrix

```

F_M=featurematrix;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Classify\featurematrix\3clfm.mat'];
F_M=load(F_M,FileName);

```

%Linear Classifier

```

PlotFeatures=[1 2];
Method=['MDC'];
P.metric=['Mahalanobis'];
TrainTestData=['50:50'];
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Classify\featurematrix\lc\3cl.mat'];
ProgressBarFlag=[0];
C_O=gBSlinearclassifier(F_M,Method,P,TrainTestData,PlotFeatures,...
FileName,ProgressBarFlag);

```

Neural Network

Neural Network allows to calculate a Multi-Layer Perceptron (MLP) or a Radial Basis Function (RBF).

Artificial Neural Networks (ANNs) can approximate the discriminant function by varying the connection strength (weight value) between the units. Such networks can have only one layer or multiple sequential layers (MLP). The MLP available in g.BSanalyze has 1 input layer, 1 hidden unit and 1 output layer.

Radial Basis Function networks have 2 layers. The first one performs a non-linear parameter transformation and the second layer makes a linear discrimination of the first layer parameters. The idea of RBF is based on the assumption that a complex pattern classification problem becomes linearly better separable when the parameters are non-linearly mapped to higher dimensionality.

The window has the following **OPTIONS** settings:

No. inputs ... number of input units. This number corresponds to the number of feature channels of the feature matrix

No. hidden units ... enter the number of hidden units

No. outputs ... number of output units. This number corresponds to the number of classes of the feature matrix

Learning rate ... specify the learning rate of the neural network. The learning rate controls the learning process. It gives the influence of the present example compared to all the past examples. A learning rate of 0 means that the neural network is not changed at all, while a learning rate of 1 would change the neural network according to the present example, independent of all previous examples.

Stop error ... enter the stop error rate

Epochs ... specify the number of training iterations

Momentum (only for MLP) ... enter the momentum value

Perform the following steps to classify a BCI experiment data-set:

1. Start **Neural Network** from the **Classification** menu and open the feature matrix file `bandpower.mat` from

`Documents\gtec\gBSanalyze\testdata\BCI\featurematrix`

The feature matrix has 4 channels, 160 trials and was created for 8 time points (1000 ms, 2000 ms, ... , 8000 ms).

Neural Network

Perform a classification of a feature matrix with a neural network. The neural net can be used for real-time analysis with g.RTsys.

Load FEATURE MATRIX:

Name of feature matrix file: `...alyze\testdata\BCI\featurematrix\bandpower.mat`

Dimension: `4 160 8` [features x trials x time points]

OPTIONS:

No. Learning Epochs:

No. hidden units: Stop Momentum:

No. outputs:

Select FEATURE CHANNELS:

Map feature no.: against:

Choose METHOD and OPTIONS:

Classification method:

Training / test-sets:

Result procedure: Show with Result2D Open classifier window Save results

Automatic treemaker is:

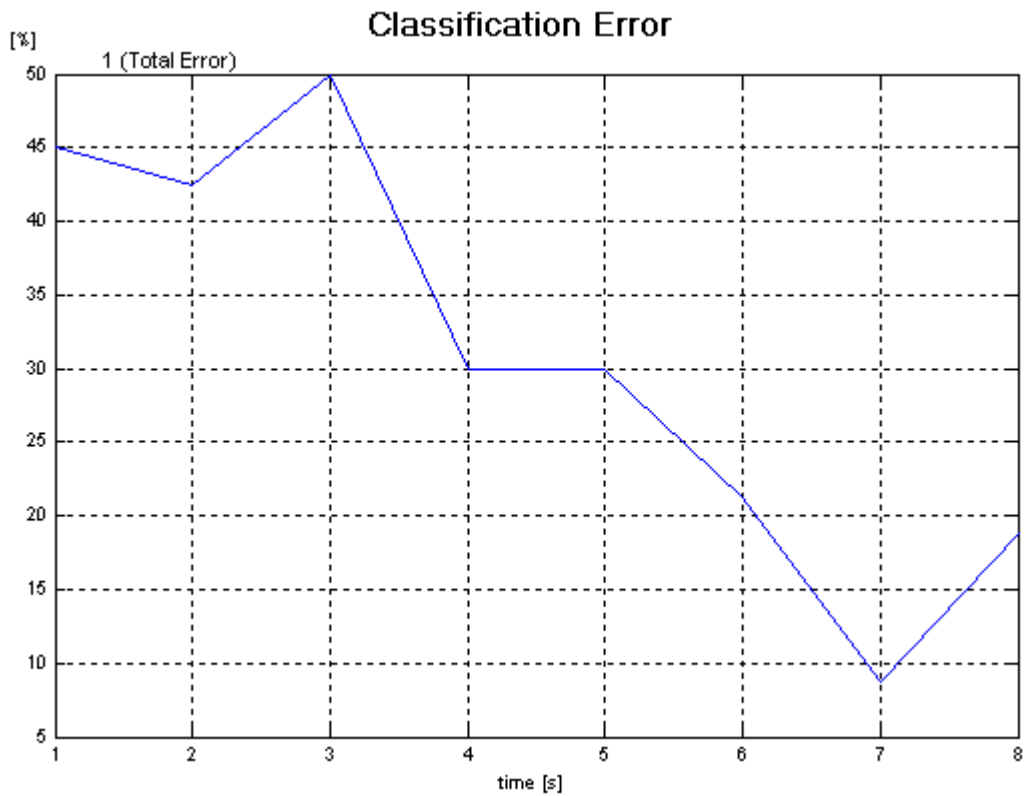
Filename: `...stdata\BCI\featurematrix\nn\NNbp.mat`

2. Set the **No. hidden units** to 8 and the **Epochs** to 4000
3. Enter 3 under **Map feature no.** and 4 under **against**
4. Select `Train 50 % - Test 50 %` to split the training and testing data in 50 % partitions

5. Check **Save results** and enter the filename `NNbp.mat`

6. Press **Start** to calculate the neural network

`gResult2d` opens automatically with the classification error time course. The minimum error of 8 % is reached at second 7.



The MATLAB Editor shows also the weight vectors of the generated MLP network. Classifier 1 shows the weights for the hidden layer (8 hidden layer nodes = 8 columns, 4 inputs layers + 1 bias = 5 rows). Classifier 2 represents the output layer (2 columns = 2 output nodes, 8 hidden layer nodes + 1 bias value = 9 rows).

```

Second/Sample: 7.000 896.000
Classifier 1
-1.211 -4.580 -1.104 0.576 0.845 0.307 0.084 0.110
-1.421 -2.675 -1.516 -1.390 1.171 0.171 -0.117 -0.947
-0.290 -1.677 -0.728 0.310 0.382 1.271 -0.422 1.423
 1.071 3.122 -0.961 0.254 0.895 4.709 0.672 -1.460
 0.781 2.537 -0.795 2.089 0.172 -0.107 -0.618 0.681
Classifier 2
 0.871 -1.360
-2.710 2.593
 1.158 -2.000
-0.977 0.010
 0.169 0.903
-0.093 -0.157
 2.087 0.033
-1.029 1.952
 0.781 0.432

```

To perform the example demonstrated above from the MATLAB command line use the following code:

```

%Load FeatureMatrix
F_M=featurematrix;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\featurematrix\bandpower.mat'];
F_M=load(F_M,FileName);

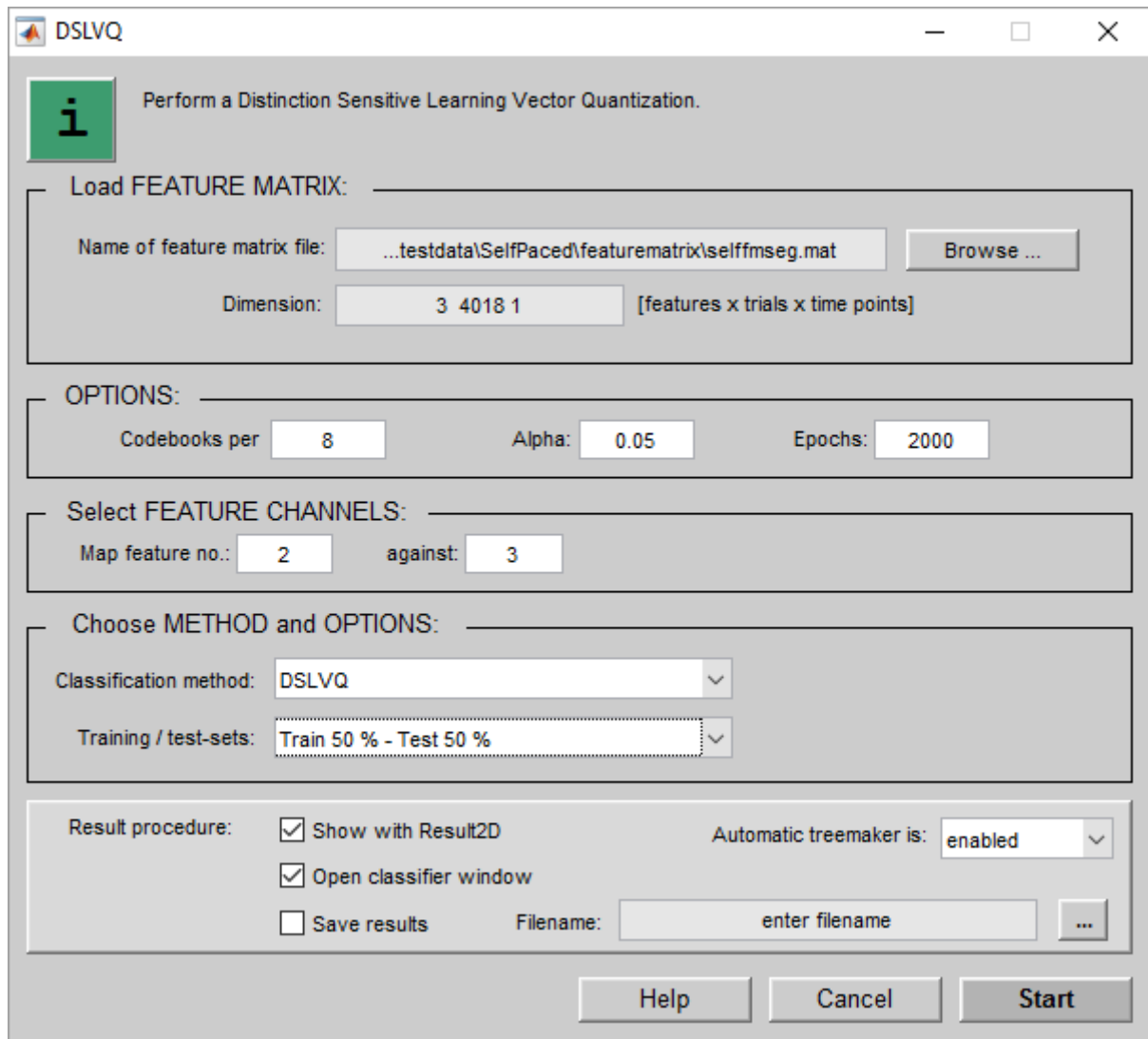
%Neural Network
P.ninput=[4];
P.nhidden=[8];
P.noutput=[2];
P.learningrate=[0.0002];
P.stoperror=[0.01];
P.epochs=[4000];
P.momentum=[0.98];
PlotFeatures=[3 4];
Method=['MLP'];
TrainTestData=['50:50'];
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\featurematrix\nn\NNbp.mat'];
ProgressBarFlag=[0];
C_O=gBSneuralnetwork(F_M,Method,P,TrainTestData,PlotFeatures,...
FileName,ProgressBarFlag);

```

DSL VQ

The method Distinction Sensitive Learning Vector Quantization is based on the individual testing of candidate feature subsets. The classifier is initially trained on all features and through implicit feature relevance analysis the system finds which features are not necessary and adapts the weighting of the features accordingly. Finally only the relevant features are used for the classification problem.

The window has the following **OPTIONS** settings:



The screenshot shows the DSLVQ software window with the following settings:

- Load FEATURE MATRIX:**
 - Name of feature matrix file:
 - Dimension: [features x trials x time points]
- OPTIONS:**
 - Codebooks per:
 - Alpha:
 - Epochs:
- Select FEATURE CHANNELS:**
 - Map feature no.: against:
- Choose METHOD and OPTIONS:**
 - Classification method:
 - Training / test-sets:
- Result procedure:**
 - Show with Result2D
 - Open classifier window
 - Save results
 - Automatic treemaker is:
 - Filename:

Buttons:

Codebooks per class ... define the number of codebooks for each class

Alpha ... define the learning speed of the algorithm

Epochs ... define the number of iterations

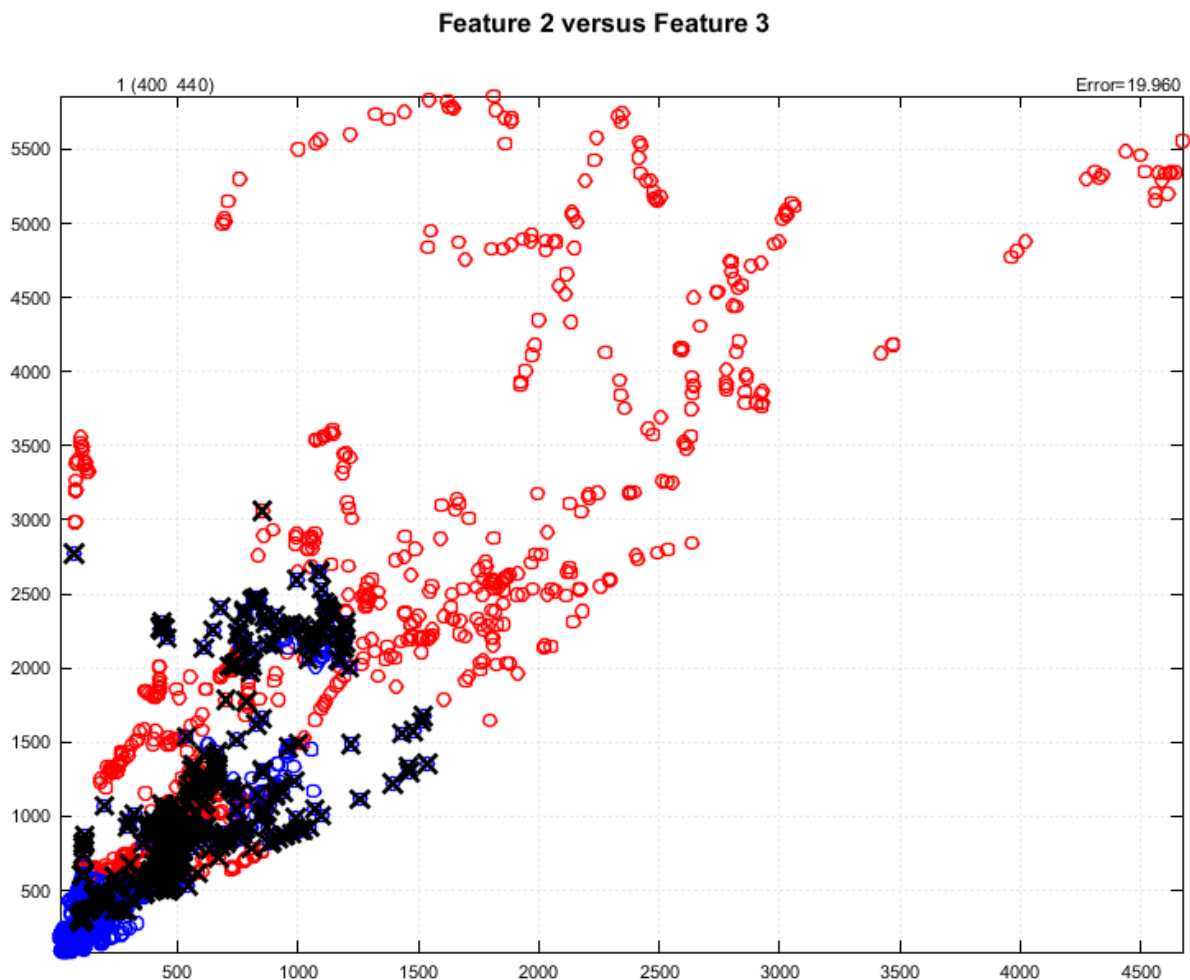
Perform the following steps to classify a finger movement ECoG experiment:

1. Open the **DSLIVQ** window from the **Classification** menu and load the feature matrix file `selffmseg.mat` from

```
Documents\gtec\gBSanalyze\testdata\SelfPaced\featurematrix
```

2. Enter under **Codebooks per class** 8 and set the number of **Epochs** to 2000
3. Enter under **Map feature no.** 2 and under **against** 3
4. Chose the `Train 50 % - Test 50 %` option
5. Press the **Start** button to train the DSLIVQ

gResult2d opens with the classification result. The classification error is 15.978 % and the plot shows the distribution of features 2 versus feature 3. Red colors correspond to the first class (segment 2000-2200 ms) and blue colors correspond to the second segment (5000 – 5200 ms). Black crosses indicate wrong classified examples.



To perform the example demonstrated above from the MATLAB command line use the following code:

```
%Load FeatureMatrix
F_M=featurematrix;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\SelfPaced\featurematrix\selffmseg.mat'
];
F_M=load(F_M,FileName);

%DSLQVQ
P.CBperclass=[8];
P.alpha=[0.05];
P.epochs=[2000];
PlotFeatures=[2 3];
Method=['DSLQVQ'];
TrainTestData=['50:50'];
FileName=[''];
ProgressBarFlag=[0];
C_O=gBSdslvq(F_M,Method,P,TrainTestData,PlotFeatures,FileName,...
ProgressBarFlag);
```

Support Vector Machine Classifier

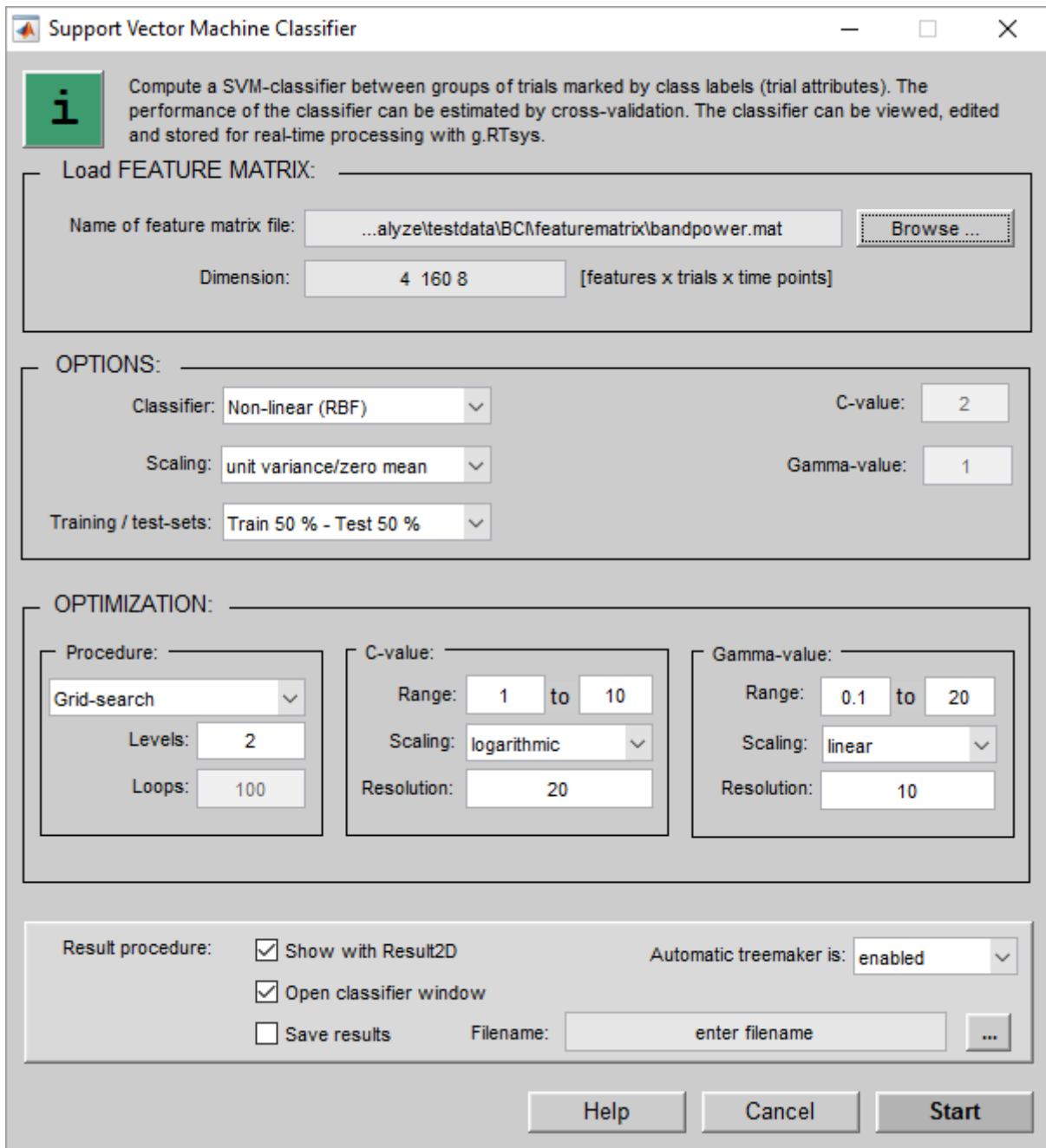
Support Vector Machine Classifier creates a classifier based on support vector machines [Cortes 1995].

To achieve good performance, it is required to scale the data appropriately, three scaling methods are implemented: **unit variance/zero mean**, **scaling to range [0 1]**, and **scaling to range [-1 1]**. Further, the hyperparameters have to be tuned for the problem. Use a linear SVM for linear problems, and tune the trade-off parameter **C-value** with **OPTIMIZATION**. For non-linear problems use a radial basis function (RBF) kernel, this adds an additional hyperparameter **Gamma-value** that can be also optimized.

Two procedures for the tuning of hyperparameters are implemented: **Grid-search** and **Random-search**. The latter is recommended if a larger search space is defined. **Grid-Search** also allows a fine tuning of the parameters over several levels. Both optimization procedures use cross-validation to determine the best parameters. The **Range**, of the tuned hyperparameters can be selected as well as the **Resolution** between the range and the **Scaling**.

References:

Cortes, C. and Vapnik, V. "Support-Vector Networks", Machine Learning, 20, pp. 273-297, 1995.



Example:

Perform the following steps to make a classification of an EEG-based brain-computer interface data-set:

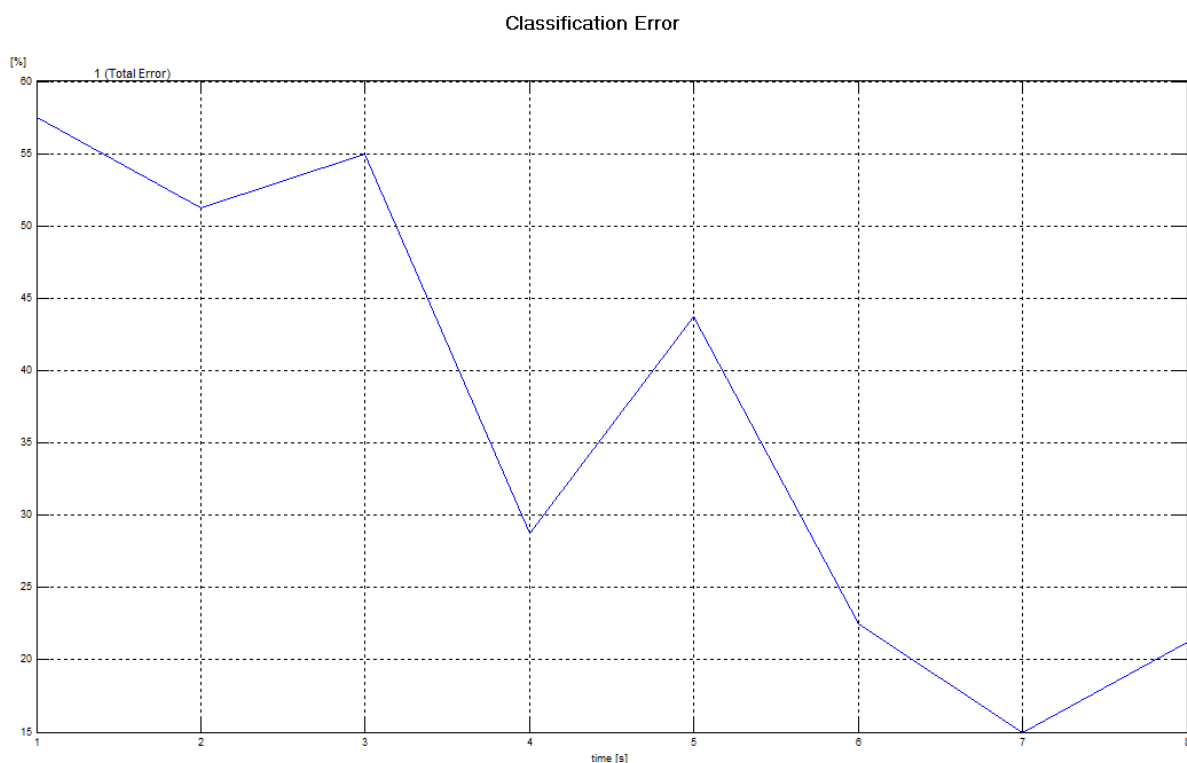
1. Open the window **Support Vector Machine Classifier** from the **Classification** menu
2. Press the **Browse** button and select the feature matrix file `bandpower.mat` that was created in the previous example and is stored under

`Documents\gtec\gBSanalyze\testdata\BCI\featurematrix`

The featurematrix contains 4 features (2 bandpower values in the alpha range and 2 bandpower values in the beta range), 160 trials (80 right and 80 left) and 8 time points (1000, 2000, ... 8000 ms)

3. Select **Non-linear (RBF)** for the Classifier to use a radial basis function as kernel
4. Select unit **variance/zero mean** as scaling method
5. Select **Train 50 % - Test 50 %** to train the classifier on 50 % of the data and test it on the other 50 %
6. Select **Grid-search** for Optimization, using **2 Levels**
7. Select for the **C-value** a **Range** between 1 and 10 with logarithmic **Scaling** and a **Resolution** of 20
8. Select for the **Gamma-value** a **Range** between 0.1 and 20 with linear **Scaling** and a **Resolution** of 10
9. Press the **Start** button

gResult2d opens with the classification result. The classification error is at the beginning around 50 - 55 % and drops down to 15 % at second 7.



The MATLAB Editor shows the ASCII description of the classifier. The kernel, the optimization method and the optimized C-values and Gamma-values for each sample are shown at the beginning:

```
Support Vector Machine
Kernel: radial basis function
```

```
Optimization method: Grid-search
Number of levels:    2
```

```
Optimization of the C-value:
Range:    [1.0 10.0]
Scaling:  logarithmic
Resolution: 20
```

```
Optimized C-values:
Second/Sample/C-value:
```

```

1.000 128.000 2.0000
2.000 256.000 2.0000
3.000 384.000 2.0000
4.000 512.000 2.0000
5.000 640.000 2.0000
6.000 768.000 2.0000
7.000 896.000 2.0000
8.000 1024.000 2.0000

```

Optimization of the Gamma-value:
Range: [0.1 20.0]
Scaling: linear
Resolution: 10

Optimized Gamma-values:
Second/Sample/Gamma-value:
1.000 128.000 1.0000
2.000 256.000 1.0000
3.000 384.000 1.2222
4.000 512.000 8.0000
5.000 640.000 1.0000
6.000 768.000 1.0000
7.000 896.000 1.0000
8.000 1024.000 1.0000

The classification error is shown in the following table: The first and second columns of the matrix show the classification time point in seconds and samples, the third column shows the error.

```

Second/Sample/Total Error/[Error Class 1/Error Class 2...]
1.000 128.000 57.5
2.000 256.000 51.2
3.000 384.000 55.0
4.000 512.000 28.7
5.000 640.000 43.8
6.000 768.000 22.5
7.000 896.000 15.0
8.000 1024.000 21.3

```

The following code shows how to perform the example demonstrated above from the MATLAB command line.

```

%Load FeatureMatrix
F_M=featurematrix;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\featurematrix\bandpower.mat'];
F_M=load(F_M,FileName);

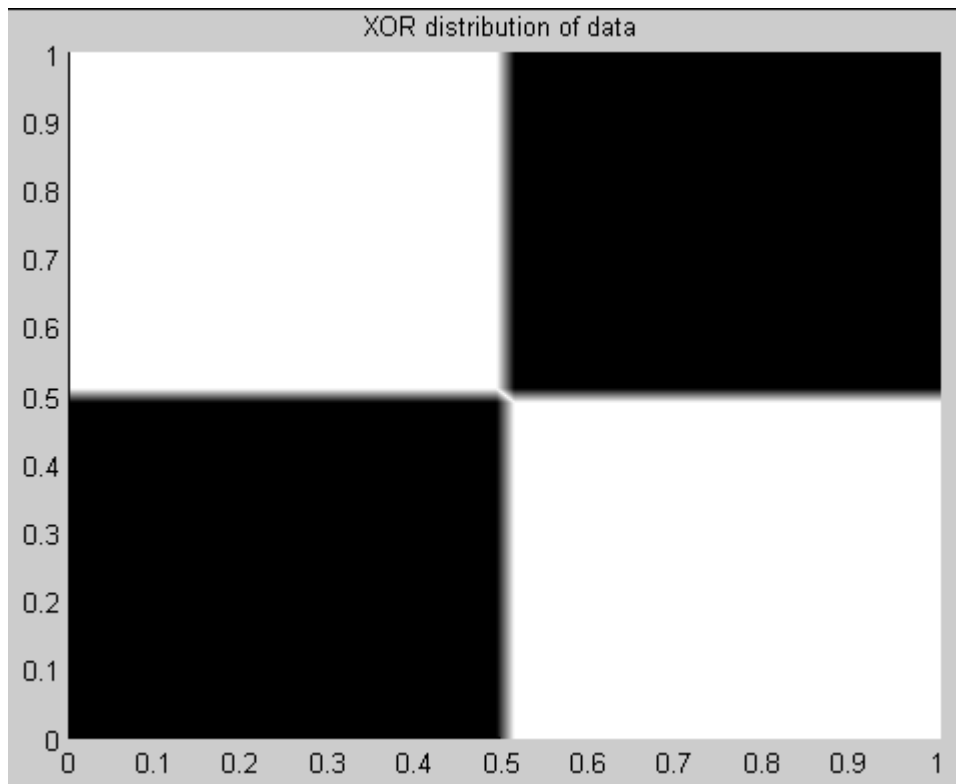
%Support Vector Machine Classifier
CLOption=[1];
Scaling=[1];
TrainTestData=['50:50'];
CParam=[10];
GParam=[0.5];
Optimization=[1];
Levels=[2];
Loops=[100];
CRange=[1.0 10];
CScaling=[1];
CResolution=[20];
GRange=[0.1 20];

```

```
GScaling=[0];
GResolution=[10];
FileName='';
ProgressBarFlag=[1];
C_O = gBSsvmclassifier(F_M,CLOption,Scaling,TrainTestData,CParam,GParam,...
Optimization,Levels,Loops,CRange,CScaling,CResolution,GRange,GScaling,...
GResolution,FileName,ProgressBarFlag);
```

Comparison of LDA and DSLVQ

This section compares the LDA and DSLVQ methods on the example of an XOR problem. The XOR distribution is demonstrated in the figure below. Basically the data-set consists of 2 channels with values between 0 and 1. If both channels have values below 0.5 or above 0.5 than the example belongs to class 0. If one channel is below 0.5 and the second channel is above 0.5 than the example belongs to class 1.



Perform the following steps:

1. Open the data-set `XOR.mat` from

`Documents\gtec\gBSanalyze\testdata\Classify`

into the Data Editor

2. Select the **Feature Matrix** window from the **Classification** menu to generate a feature matrix for a single time point.

Feature Matrix

Generate a feature matrix as input for the classification methods. Select class allows to select trials with a certain attribute. Each attribute corresponds to a class. Select time point allows to select specific time points. Each time point corresponds to a class.

Specify CLASSIFICATION INTERVAL:

Start at: 600 [ms] Step: 600 [ms] Stop at: 600 [ms] Merge time points
 3 [samples] 3 [samples] 3 [samples]

Specify CLASS LABELS / TIME POINT:

Select class: ARTIFACT REMOVE
 1
 0

Select time point: 600

Select FEATURE CHANNELS:

Choose METHOD and OPTIONS:

Classification method: Linear Discriminant Analysis (LDA)
 Randomly permutate the matrix

Result procedure: Classify data Automatic treemaker is: enabled
 Save results: Filename: ...data\Classify\featurematrix\xorfm.mat

3. Set **Start at**, **Step** and **Stop at** to 600 ms and select classes 1 and 0 under **Select class**
4. Check the **Save results** box and enter the filename `xorfm.mat`
5. Press **Start** to save the feature matrix file

6. Open the **Linear Classifier** window under **Classification** and **Browse** for the feature matrix file `xorfm.mat`

Linear Classifier

Compute a classifier (weight vector) between groups of trials marked by class labels (trial attributes). The performance of the classifier can be estimated by cross-validation. The weight vector can be viewed, edited and stored for real-time processing with q.RTsys.

Load FEATURE MATRIX:

Name of feature matrix file:

Dimension: [features x trials x time points]

Select FEATURE CHANNELS:

Map feature no.: against:

Choose METHOD and OPTIONS:

Classification method: Metric Mahalanobis distance

Training / test-sets: Euclidian distance

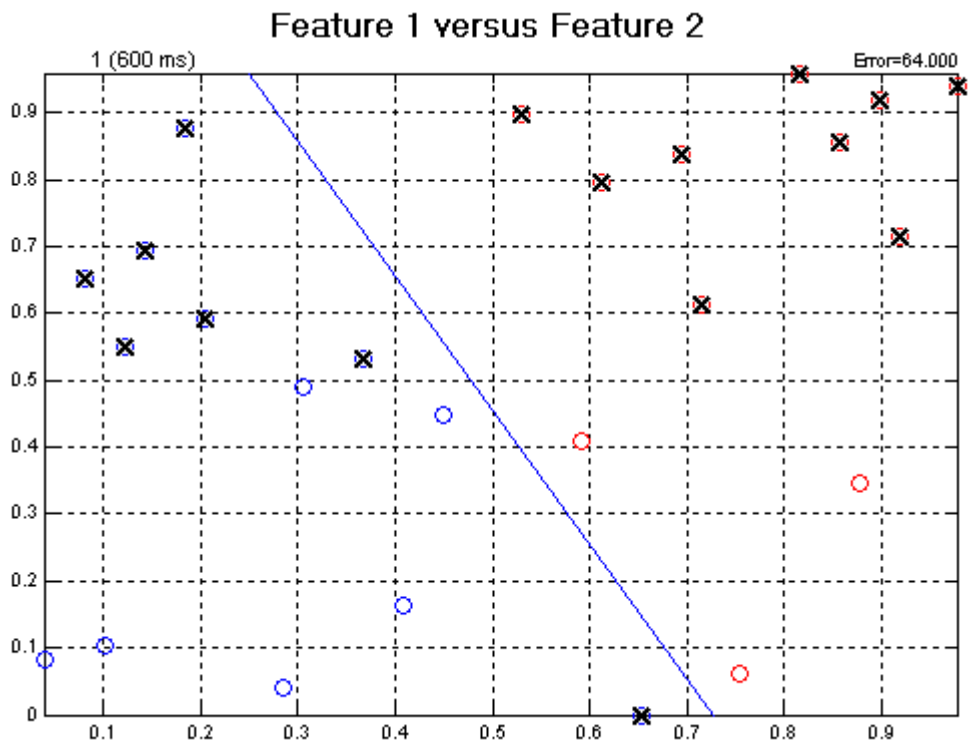
Result procedure: Show with Result2D Automatic treemaker is:

Open classifier window

Save results Filename:

7. Select `Train 50 % - Test 50 %` and press the **Start** button

gResult2d maps feature 1 versus feature 2 and shows the classification error of 64 %. The LDA is not able to discriminate the XOR problem.



8. Close the **Linear Classifier** window

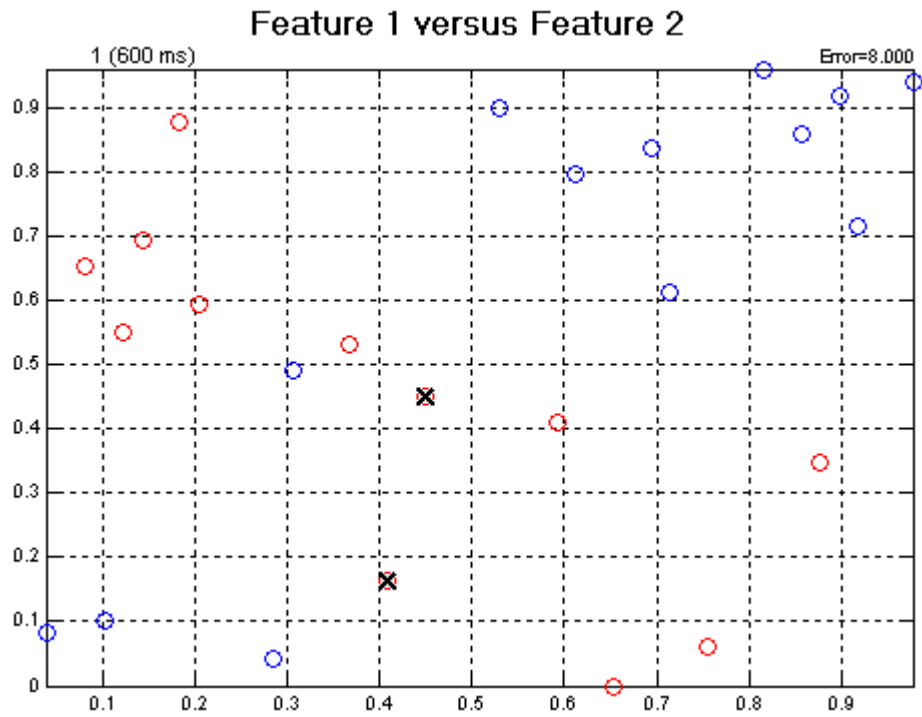
9. Open the **DSL**VQ window
10. Load again the `xorfm.mat` file and set the **Codebooks per class** to 8 and the **Epochs** to 4000
11. Select `Train 50 % - Test 50 %` and press the **Start** button

The screenshot shows the DSLVQ software window with the following configuration:

- Load FEATURE MATRIX:**
 - Name of feature matrix file: `...lyzeltestdata\Classify\featurematrix\xorfm.mat` (with a `Browse ...` button)
 - Dimension: `2 50 1` [features x trials x time points]
- OPTIONS:**
 - Codebooks per: `8`
 - Alpha: `0.05`
 - Epochs: `4000`
- Select FEATURE CHANNELS:**
 - Map feature no.: `1` against: `2`
- Choose METHOD and OPTIONS:**
 - Classification method: `DSL`VQ
 - Training / test-sets: `Train 50 % - Test 50 %`
- Result procedure:**
 - Show with Result2D
 - Open classifier window
 - Save results
 - Automatic treemaker is: `enabled`
 - Filename: `enter filename` (with a `...` button)

At the bottom of the window are three buttons: `Help`, `Cancel`, and `Start`.

gResult2d shows now the classification result of the DSLVQ classifier with an error rate of 8 %. Note that only 2 trials (which are close to the border of 0.5) were wrongly classified. If the number of training examples is enhanced the error rate is reduced.



To perform the example demonstrated above from the MATLAB command line use the following code:

%Load Data

```
P_C=data;
File= ['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Classify\XOR.mat'];
P_C=load(P_C,File);
```

%Feature Matrix

```
Interval=[3 3 3];
AttributeName={
    '1'
    '0'
};
ChannelExclude=[];
Permutate=0;
MergeTimePoints=0;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Classify\featurematrix\xorfm.mat'];
ProgressBarFlag=[0];
F_O=gBSfeaturematrix(P_C,Interval,AttributeName,Permutate,...
MergeTimePoints,ChannelExclude,FileName,ProgressBarFlag);
```

%Load FeatureMatrix

```
F_M=featurematrix;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Classify\featurematrix\xorfm.mat'];
F_M=load(F_M,FileName);
```

%Linear Classifier

```
PlotFeatures=[1 2];
Method=['LDA'];
P.metric=[''];
TrainTestData=['50:50'];
FileName=[''];
ProgressBarFlag=[0];
C_O=gBSlinearclassifier(F_M,Method,P,TrainTestData,PlotFeatures,...
FileName,ProgressBarFlag);
```

%Load FeatureMatrix

```
F_M=featurematrix;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Classify\featurematrix\xorfm.mat'];
F_M=load(F_M,FileName);
```

%DSLQVQ

```
P.CBperclass=[8];
P.alpha=[0.05];
P.epochs=[4000];
PlotFeatures=[1 2];
Method=['DSLQVQ'];
TrainTestData=['50:50'];
FileName=[''];
ProgressBarFlag=[0];
C_O=gBSdslvq(F_M,Method,P,TrainTestData,PlotFeatures,FileName,...
ProgressBarFlag);
```

Receiver Operator Curve

The sensitivity and specificity of a diagnostic test depends on more than just on the quality of the test. It depends also on the definition of what constitutes an abnormal test. In practice a threshold is selected to distinguish e.g. normal from disease. The threshold level determines the number of true positives, true negatives, false positives and false negatives. By moving the threshold to a higher level the sensitivity can be improved which makes the criterion for a positive test less strict. The specificity can be improved by moving the threshold to a lower value which makes the criterion for a positive test more strict. Thus, there is a tradeoff between sensitivity and specificity. Therefore, a Receiver Operator Characteristic curve (ROC) can be used to find the optimal threshold. The ROC curve plots the true positive rate against the false positive rate for different thresholds.

The ROC curve demonstrates the following:

- It shows the tradeoff between sensitivity and specificity - an increase of the sensitivity is accompanied by a decrease of the specificity
- The test is less accurate if the ROC curve comes closer to the 45 degree diagonal line
- The test is more accurate if the curve comes close to the left and upper border
- The area under the curve is a measure of accuracy:

0.9 - 1	excellent
1.8 - 0.9	good
0.7 - 0.8	fair
0.6 - 0.7	poor
0.5 - 0.6	fail

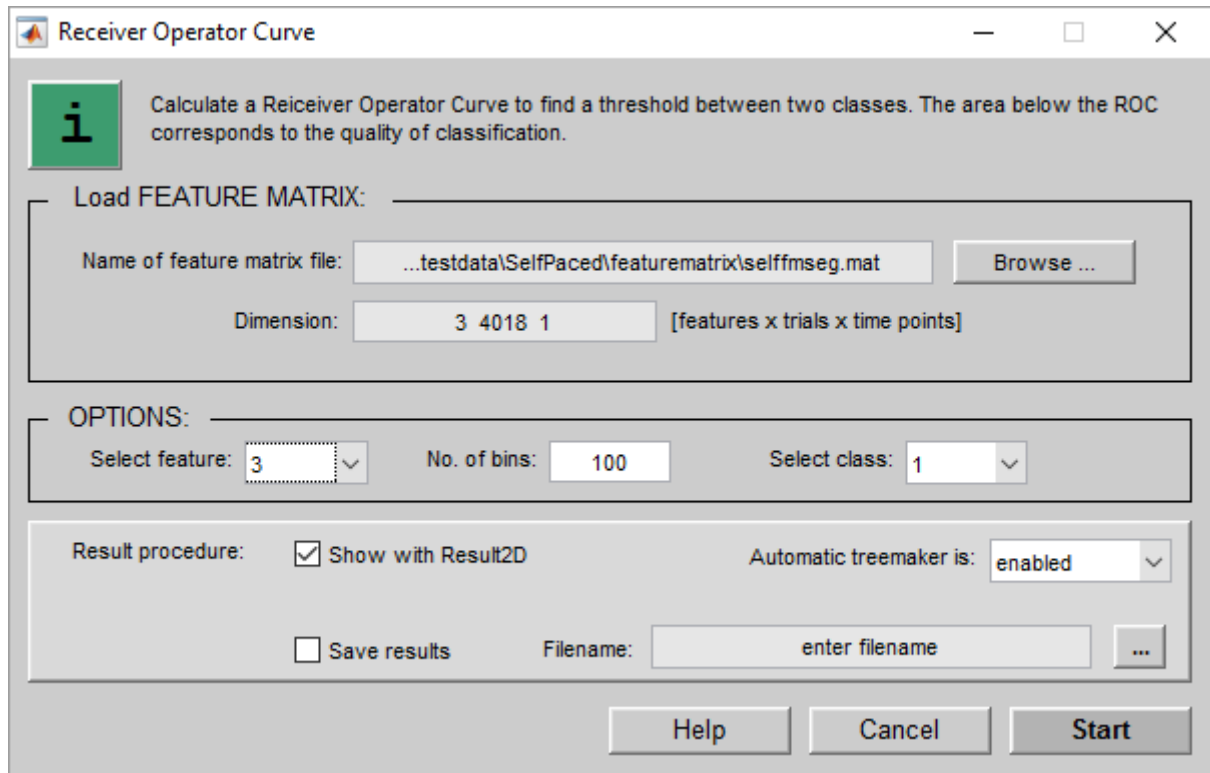
Perform the following steps:

1. Open the **Receiver Operator Curve** window from the **Classification** menu

2. Press the **Browse** button to load feature matrix file `selffmseg.mat` from

`Documents\gtec\gBSanalyze\testdata\SelfPaced\featurematrix`

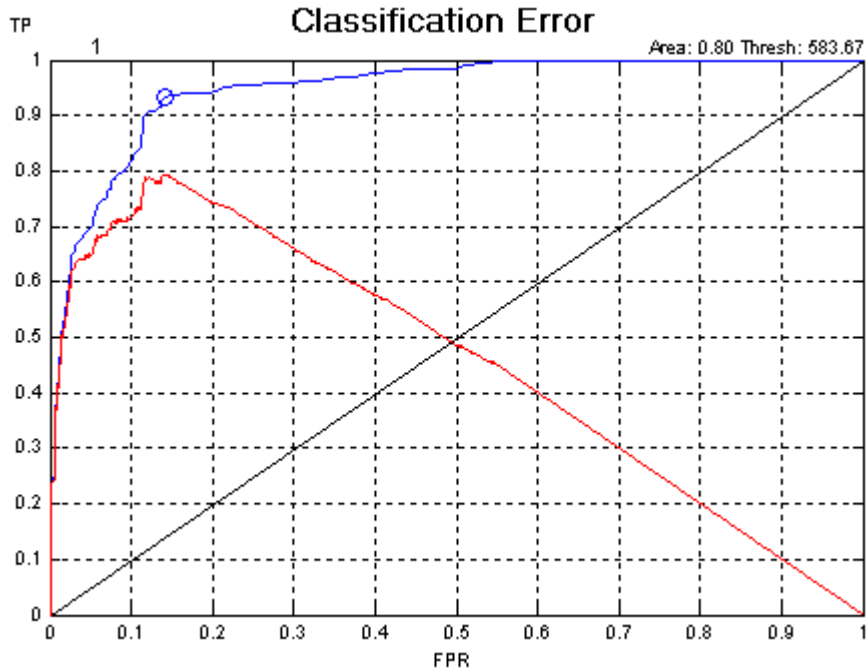
The features matrix has 3 channels and 4018 trials



3. Select feature 3

4. Press the **Start** button

gResult2d opens with the ROC curve. The true positive rate (TP) is plotted on the y-axis, the false positive rate (FPR) is plotted on the x axis. The area under the blue ROC curve is 0.8 which corresponds to a good accuracy and the optimal threshold level is 583.67 (indicated by the circle). The circle marks also the maximum of the red HF (hit-false) difference curve.



To perform the example demonstrated above from the MATLAB command line use the following code:

```
%Load Feature Matrix
F_O=featurematrix;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\SelfPaced\featurematrix\selffmseg.mat'
];
F_O=load(F_O,FileName);

%Receiver Operator Curve
FeatureNumber=[3];
NrBins=[100];
ClassNumber=[1];
FileName=[''];
ProgressBarFlag=[0];
D_O=gBSreceiveroperatorcurve(F_O,FeatureNumber,ClassNumber,...
NrBins,FileName,ProgressBarFlag);
```


DSLQV Feature Weighting

DSLQV can be used to analyze the importance of specific features to a discrimination task.

The window has the following OPTIONS settings:

Codebooks per class ... define the number of codebooks for each class

Alpha ... define the learning speed of the algorithm

Epochs ... define the number of iterations

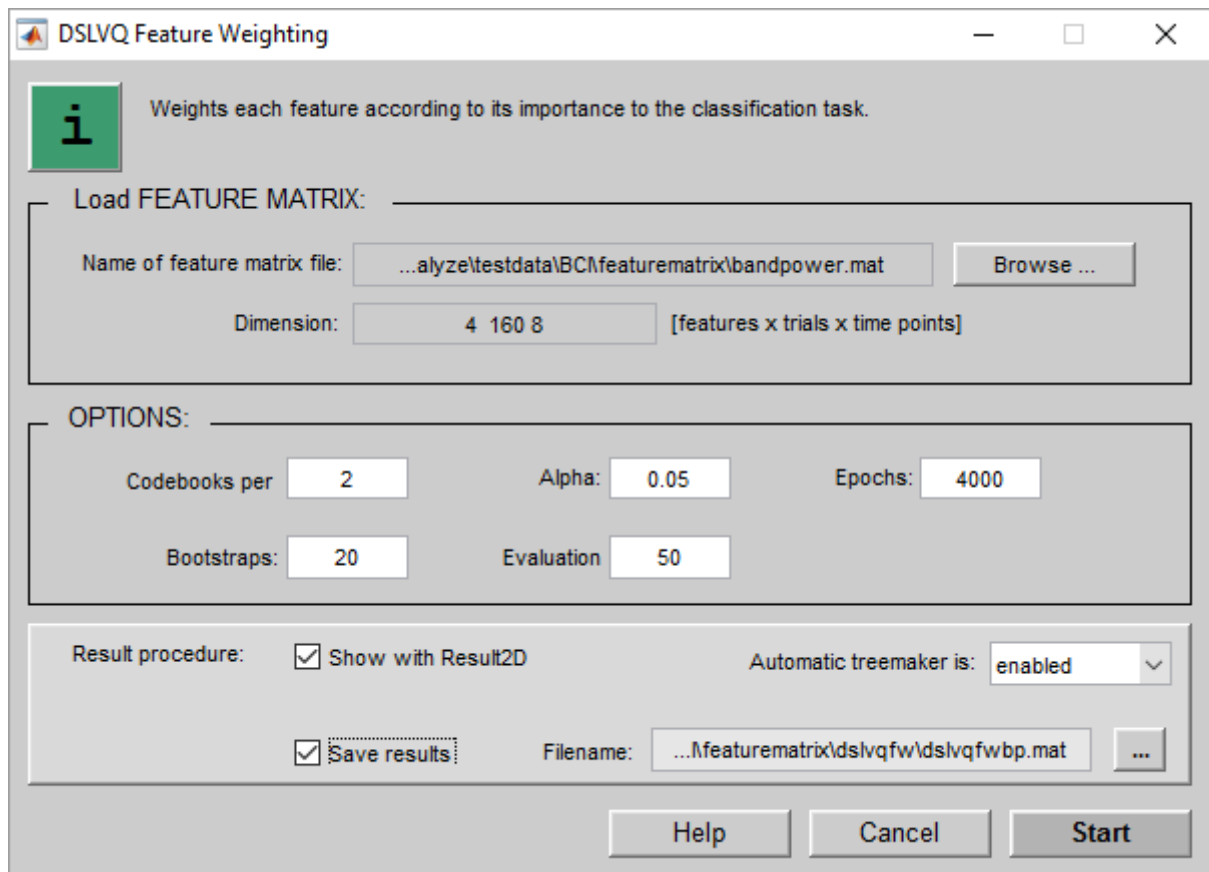
Bootstraps ... number of bootstrap repetition

Evaluation ... percentage of data used for testing'

Perform the following steps:

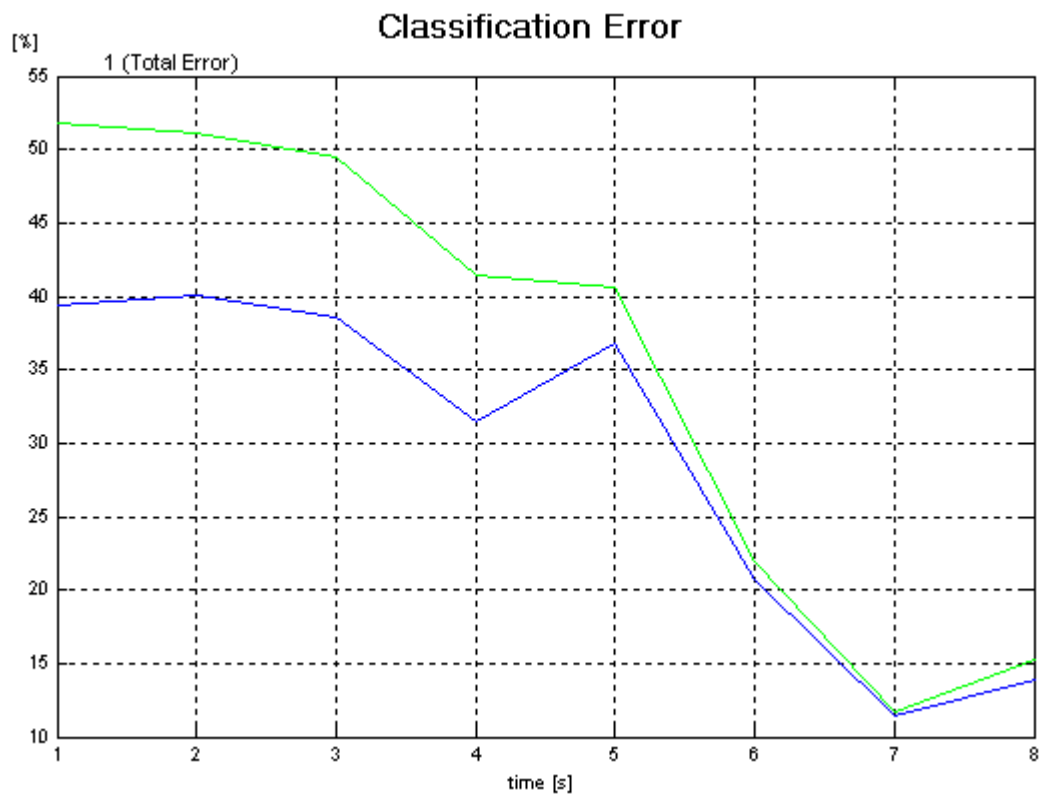
1. Open **DSLQV Feature Weighting** from the **Classification** menu and **Browse** for the feature matrix `bandpower.mat` under

`Documents\gtec\gBSanalyze\testdata\BCI\featurematrix`



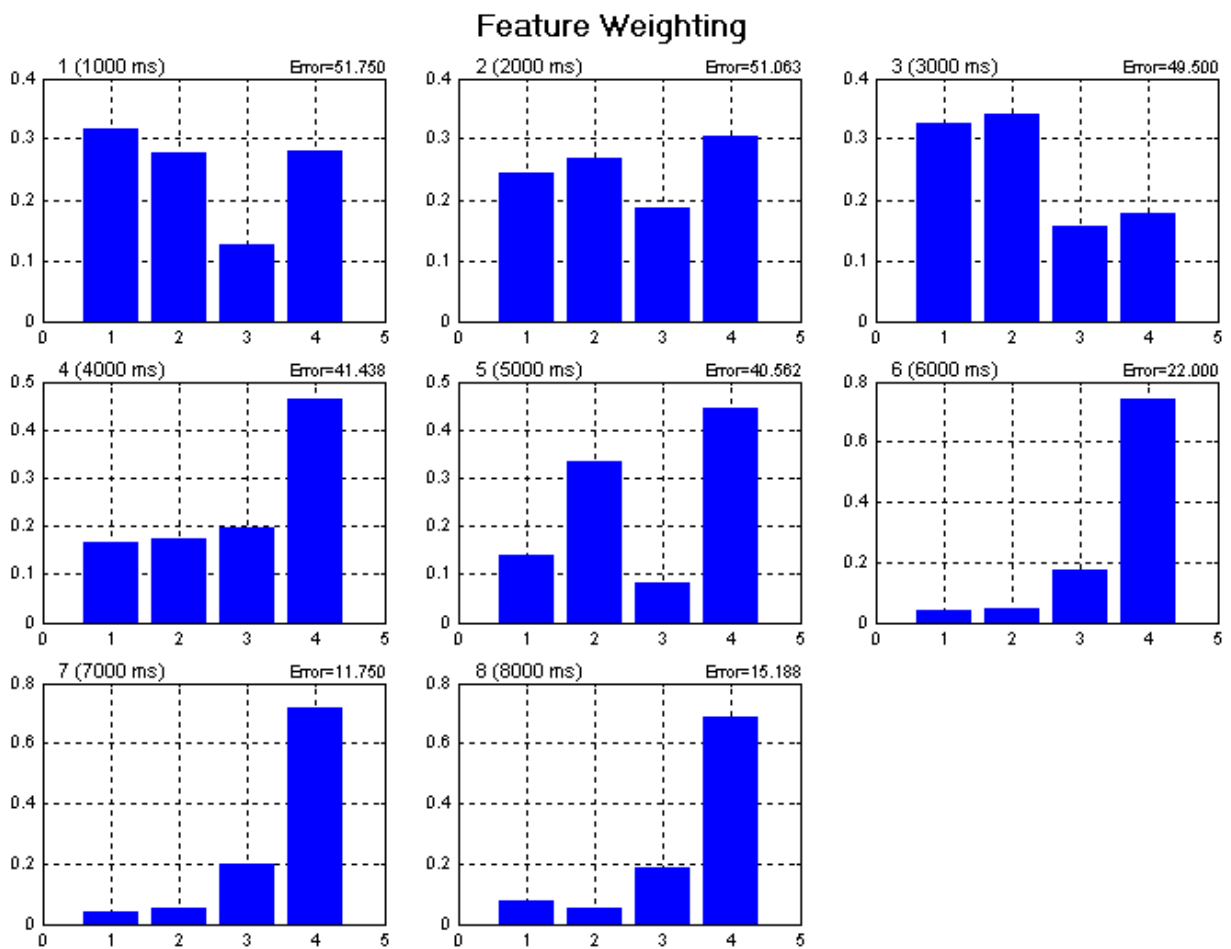
2. Set the number of **Epochs** of 4000
3. Check **Save results** and enter the filename `ds1vqfwbp.mat`
4. Press **Start** to perform the feature weighting

Page 1 of gResult2d show the classification error of the training data (blue line) and of the testing data (green line). The minimum is reached at second 7. During the first 5 seconds the error rate of the testing data is higher as the training error.



The second page of gResult2d shows the feature weights for each time point. From second 1 to 3 the error is around 50 % and drops down to a minimum of 11.75 % at second 7.

The blue bars represent the importance of each feature to the discrimination task. At second 1 features 1 is the most important one, followed by 4, 2 and 3. But the classification error is 51,75 % and therefore the result is random. At second 7 the error is 11,75 % and therefore the feature weighting can be considered as reliable. Therefore, feature 4 (bandpower in the beta range of channel 2) is the most important one. The bar of feature 3 is much smaller but the feature can still be considered as important for the classification task. Features 1 and 2 are not important and should not be considered for the discrimination.



To perform the example demonstrated above from the MATLAB command line use the following code:

```
%Load FeatureMatrix
F_M=featurematrix;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\featurematrix\bandpower.mat'];
F_M=load(F_M,FileName);

%DSLVQ Feature Weighting
P.CBperclass=[2];
P.alpha=[0.05];
P.epochs=[4000];
P.bootstraps=[20];
P.evaluation=[50];
Method=['DSLVQ'];
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\featurematrix\dslvqfwbp.mat'];
ProgressBarFlag=[0];
C_O=gBSdslvqfeatureweighting(F_M,Method,P,FileName,ProgressBarFlag);
```

KMEANS Clustering

A very common method to find the optimal position of codebook vectors is k-means. The codebook vectors are approximated iteratively.

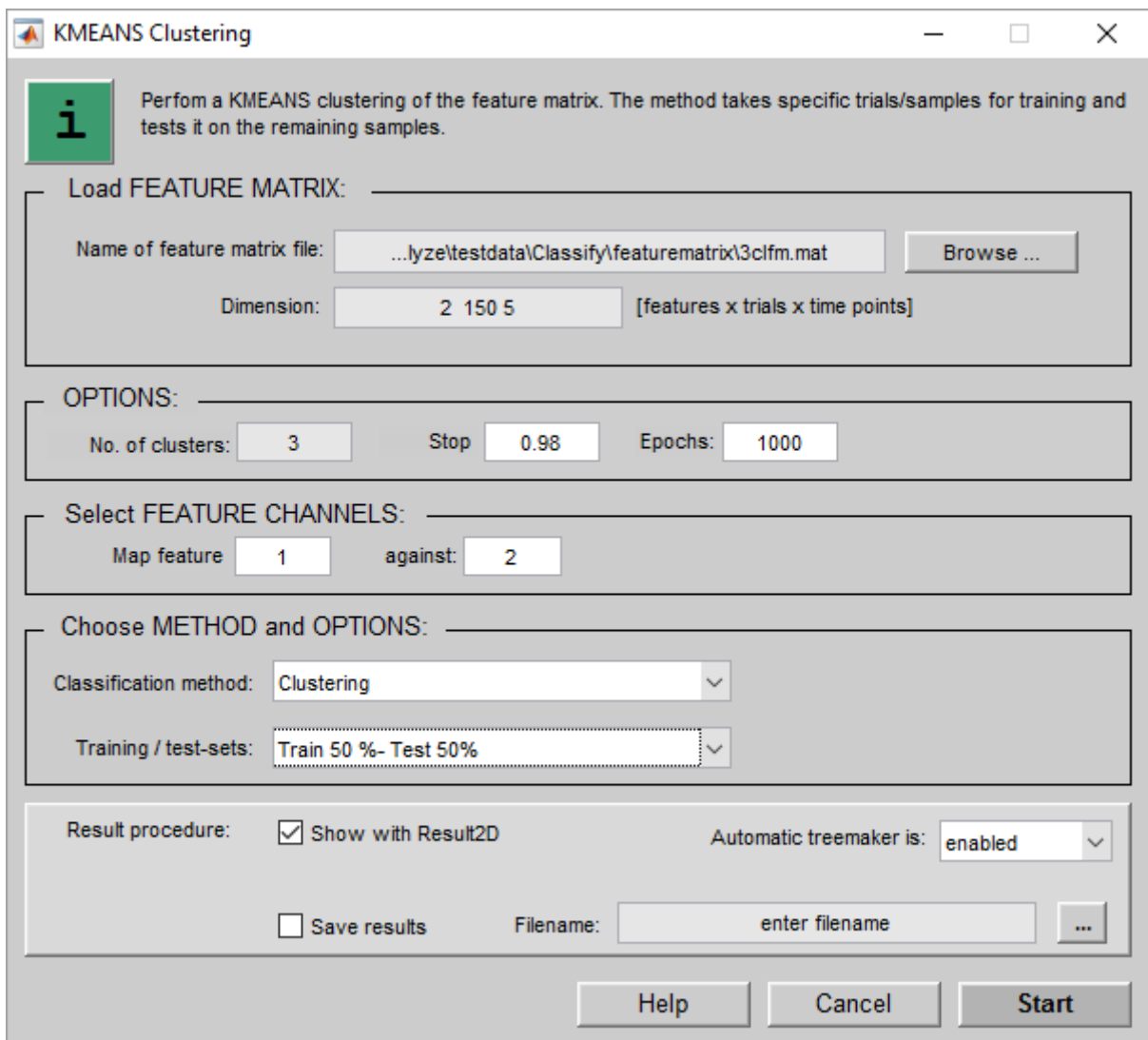
Perform the following steps to perform an unsupervised clustering of the data:

1. Open **KMEANS Clustering** from the **Classification** menu
2. Click on the **Browse** button and search for the feature matrix file `3clf.m.mat` which is stored under

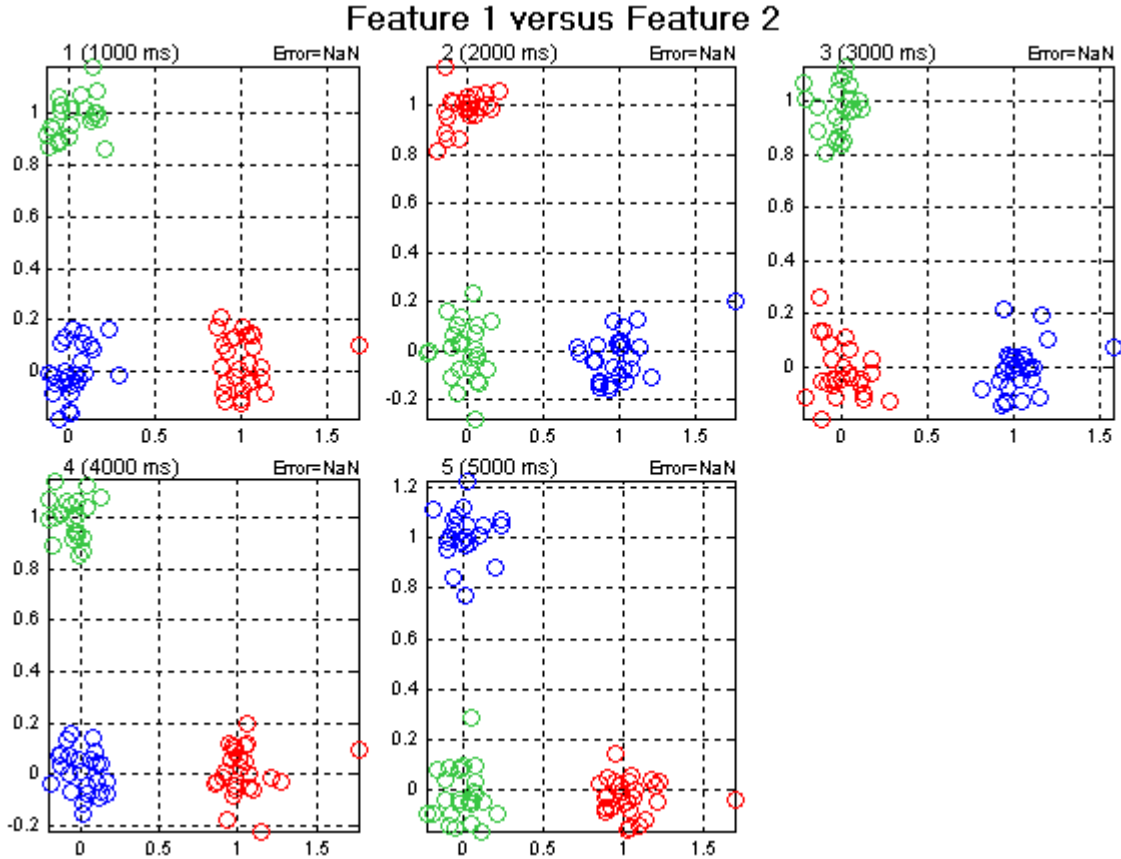
`Documents\gtec\gBSanalyze\testdata\Classify\featurematrix`

The feature matrix contains 2 feature channels with 150 examples and 3 classes. The feature matrix was calculated for 5 time points. The **No. of clusters** is set to the number of loaded classes.

3. Select `Train 50 % - Test 50 %` and press the **Start** button



gResult2d shows the clustering results for all 5 time points. The method used the first 50 % of the data for finding the codebook vectors and used the result to cluster the test data. Basically the k-means algorithm was able to find un-supervised the 3 classes. The outlier (trial 128) yielded to a wrong clustering result.



To perform the example demonstrated above from the MATLAB command line use the following code:

```
%Load FeatureMatrix
F_M=featurematrix;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\Classify\featurematrix\3clfm.mat'];
F_M=load(F_M,FileName);

%KMEANS Clustering
P.ncluster=[3];
P.stoperror=[0.98];
P.epochs=[1000];
PlotFeatures=[1 2];
Method=['KMEANS'];
TrainTestData=['50:50'];
FileName=[''];
ProgressBarFlag=[0];
C_O=gBskmeansclustering(F_M,Method,P,TrainTestData,PlotFeatures,...
FileName,ProgressBarFlag);
```

Using the Classifier

After generating a classifier with the linear or non-linear methods it is possible to use this classifier for the classification of new data.

There are two ways:

[Apply Classifier](#) – classifies data of the Data Editor with the classifier and generates a new channel in the Data Editor. This channel represents the output of the classification method.

[Test Classifier](#) – test the generated classifier on new data and produce an error rate curve and feature cloud in gResult2d

Apply Classifier

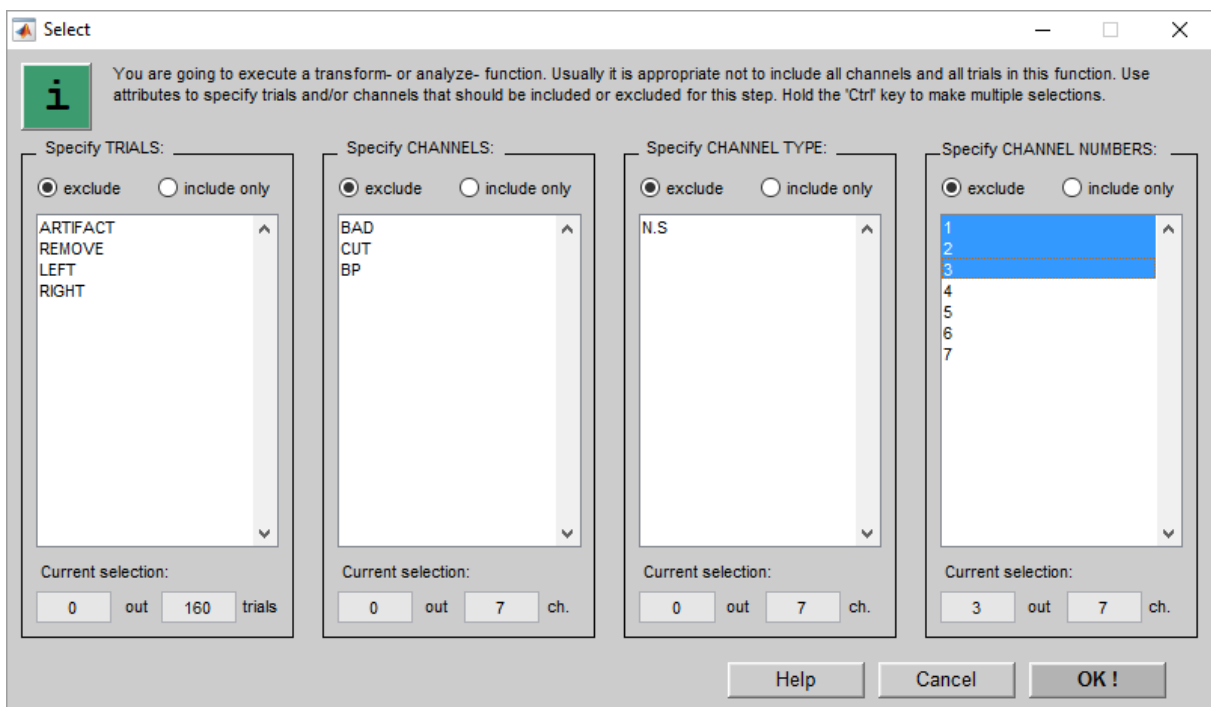
Perform the following steps:

1. Load the data-set `session1234bp.mat` from

`Documents\gtec\gBSanalyze\testdata\BCI`

into the Data Editor. The Data Editor visualizes 2 EEG channels, 1 trigger channel and 4 bandpower channels.

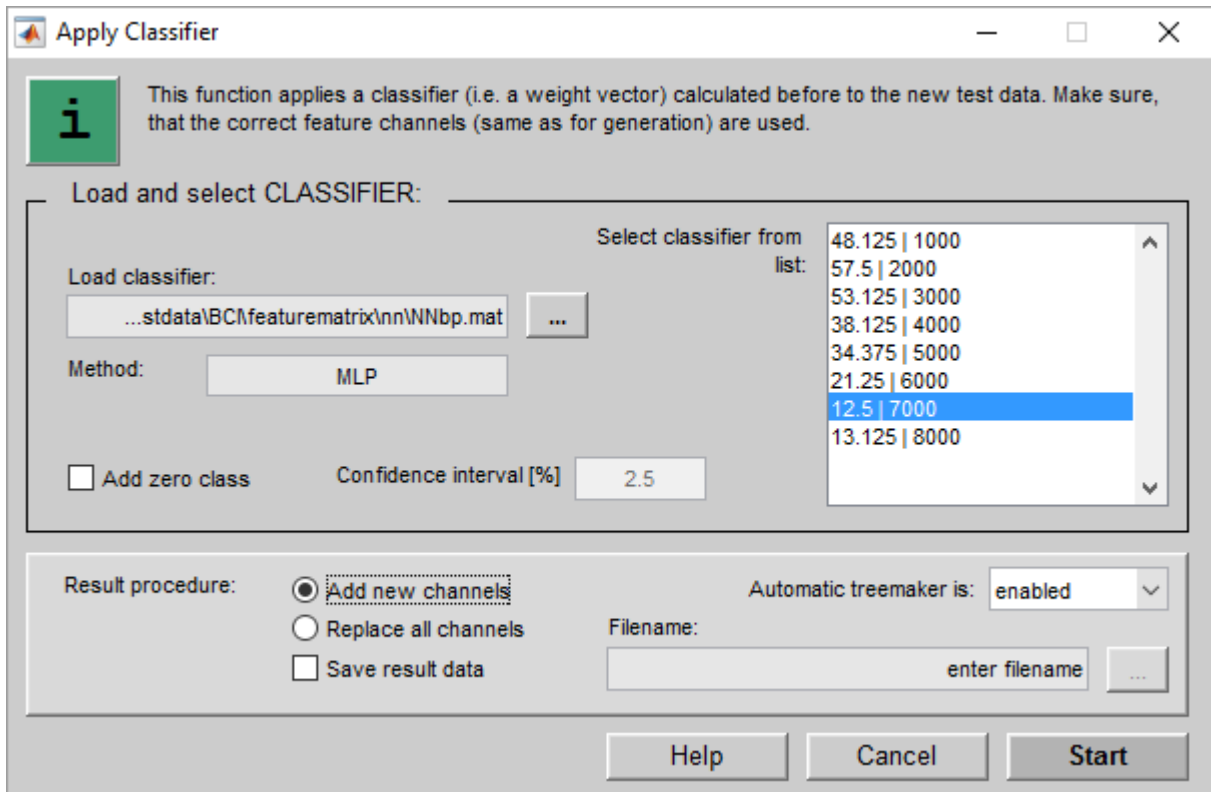
2. Open the **Cut Trials Channels** window from the **Transform** menu and exclude channels 1, 2 and 3



3. Open **Apply Classifier** from the **Classification** menu and **Browse** for the classifier file `NNbp.mat` under

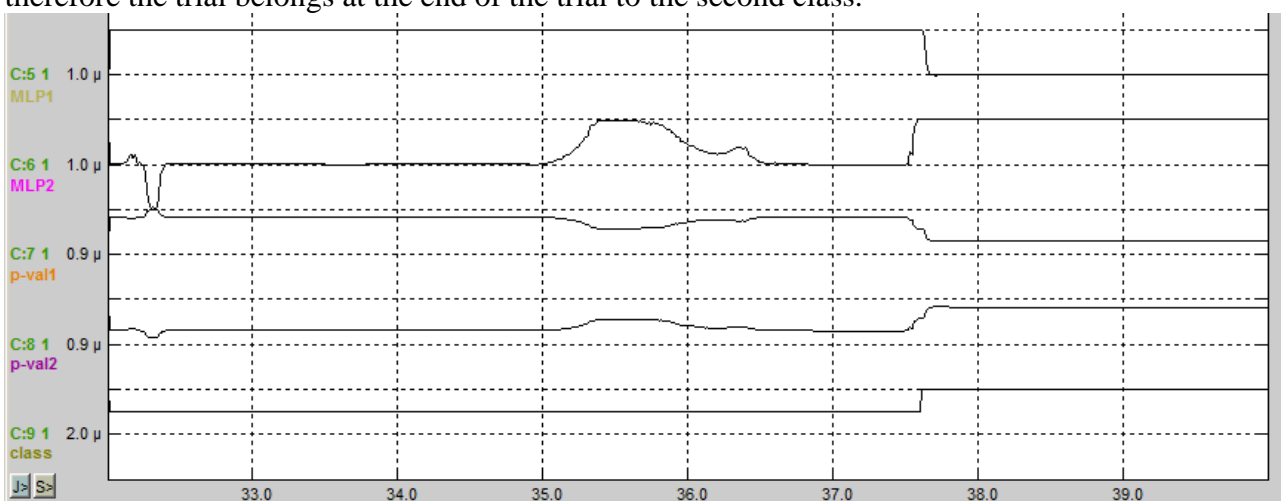
`Documents\gtec\gBSanalyze\testdata\BCI\featurematrix\mn`

The listbox shows all classification time points with the corresponding classification error.



4. Select the appropriate classifier (normally the best one) and check **Add new channels**
5. Press **Start** to classify the data

After classification the Data Editor shows five additional channels with the classification result of the neural network. The first two channels display the two outputs of the neural network. Channel 5 reaches at the end of the trial zero, channel 6 reaches at the end 1, therefore the trial belongs at the end of the trial to the second class.

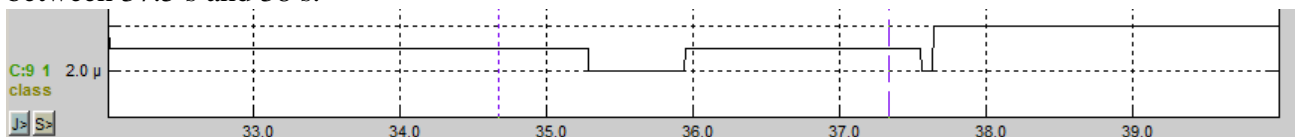


Channel 7 and 8 represent the probabilities that the data would be assigned to one of the two classes. A value of 0 means the data does not belong to the corresponding class and a value of 1 means that it belongs to the corresponding class. From channel 7 it can be deduced that it is up to 80% probable that the trial belongs at the beginning to class 1 and with 72% to the

second class at the end of the trial. This fact is also indicated by channel 9 added by the **ApplyClassifier** function. A value of 1 means that this actual time point belongs to the first class and a 2 means that the time point belongs to the second class.

- Repeat steps 1 to 5 but check the **Add zero class** and select a value of 40 for the **Confidence interval [%]** parameter

Now the channel 9 indicating the class assignment shows an additional zero class. The **ApplyClassifier** function assigns the data samples to this virtual class whenever the probability that the selected class assignment is wrong is higher than the value of the **Confidence interval [%]** parameter which was set to 40 %. This is the case in the middle of the trial between 35 s and 36 s and when the class assignment switches from class 1 to class to between 37.5 s and 38 s.



To perform the example demonstrated above from the MATLAB command line use the following code:

```
%Load Data
P_C=data;
File=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\session1234bp.mat'];
P_C=load(P_C,File);

%Select Trials and Channels
trial_id=[];
channel_id=[];
type_id=[];
channelnr_id=[1 2 3];
flag_tr='tr_exc';
flag_ch='ch_exc';
flag_type='type_exc';
flag_nr='nr_exc';
[TrialExclude, ChannelExclude]=gBSselect(P_C,trial_id,flag_tr,...
channel_id,flag_ch,type_id,flag_type,channelnr_id,flag_nr);
P_C=gBScuttrialschannels(P_C,TrialExclude,ChannelExclude);

%Load Classifier
C_O_S=classifierobj;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\featurematrix\nn\NNbp.mat'];
C_O_S=load(C_O_S,FileName);

%Apply Classifier
ClassifierNumber=[7];
Replace=['add channels'];
FileName='';
ProgressBarFlag=[0];
ConfidenceInterval=[];
P_C=gBSapplyclassifier(P_C,C_O_S,ClassifierNumber,Replace, ...
ConfidenceInterval,FileName,ProgressBarFlag );
```

Test Classifier

After calculating a classifier it is possible to test the classifier on new data.

Perform the following steps:

1. Open **Test Classifier** from the **Classification** menu and in the **Load FEATURE MATRIX** section **Browse** to the feature matrix `bandpower.mat` from

`Documents\gtec\gBSanalyze\testdata\BCI\featurematrix`

2. In the **Load and select CLASSIFIER** section **Browse** for the classifier file `NNbp.mat` under

`Documents\gtec\gBSanalyze\testdata\BCI\featurematrix\mn`

and select the classifier from second 7

3. Press the **Start** button

Test Classifier

Test a classifier on a specific feature matrix.

Load FEATURE MATRIX:

Name of feature matrix file:

Dimension: [features x trials x time points]

Load and select CLASSIFIER:

Load classifier:

Method:

Select best classifier from list:

48.13 %	1000.00 ms
57.50 %	2000.00 ms
53.13 %	3000.00 ms
38.13 %	4000.00 ms
34.38 %	5000.00 ms
21.25 %	6000.00 ms
12.50 %	7000.00 ms
13.13 %	8000.00 ms

Select FEATURE CHANNELS:

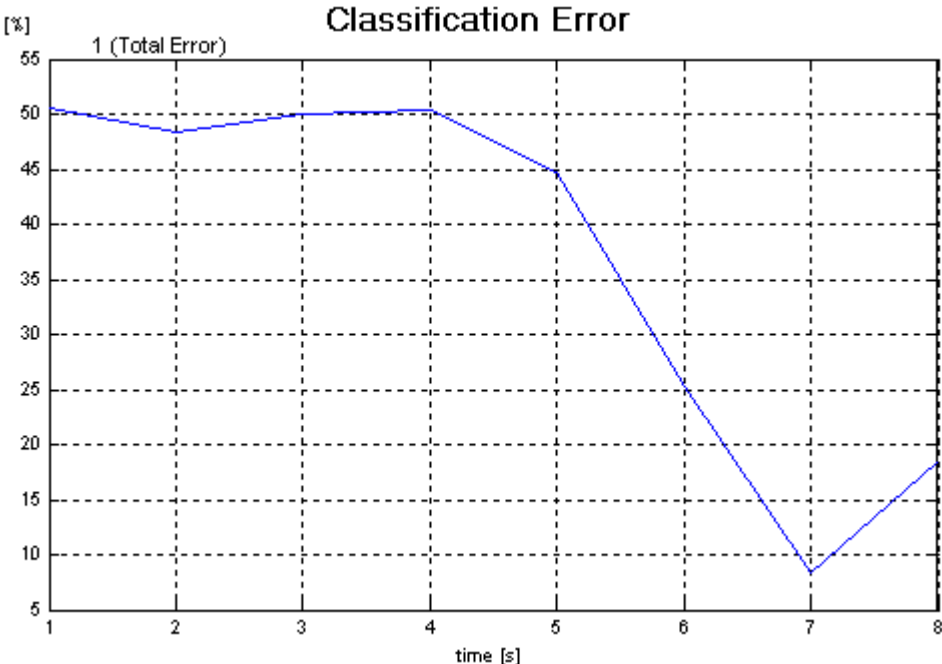
Map feature no.: against:

Result procedure: Show with Result2D Open validation window Save results

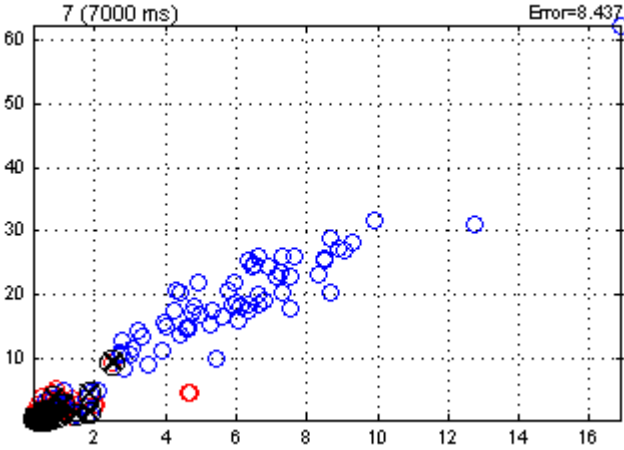
Automatic treemaker is:

Filename:

gResult2d display the time course of the classification error. The minimum is reached at second 7 and is 8 %.



Page 2 of gResult2d displays the feature map of feature 3 versus feature 4.



To perform the example demonstrated above from the MATLAB command line use the following code:

```
%Load Feature Matrix
F_O=featurematrix;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\featurematrix\bandpower.mat'];
F_O=load(F_O,FileName);

%Load Classifier
C_O=classifierobj;
FileName=['C:\Users\' getenv('USERNAME')
'\Documents\gtec\gBSanalyze\testdata\BCI\featurematrix\nn\NNbp.mat'];
C_O=load(C_O,FileName);

%Test Classifier
ClassifierNumber=[7];
PlotFeatures=[3 4];
FileName='';
ProgressBarFlag=[0];
C_O=gBStestclassifier(F_O,C_O,ClassifierNumber,PlotFeatures,FileName,...
ProgressBarFlag);
```

Classification Output Mapping

After applying a classifier to a dataset, it is possible to plot the classification outputs and calculate the classification error rates averaged over trials for the selected classes and channels.

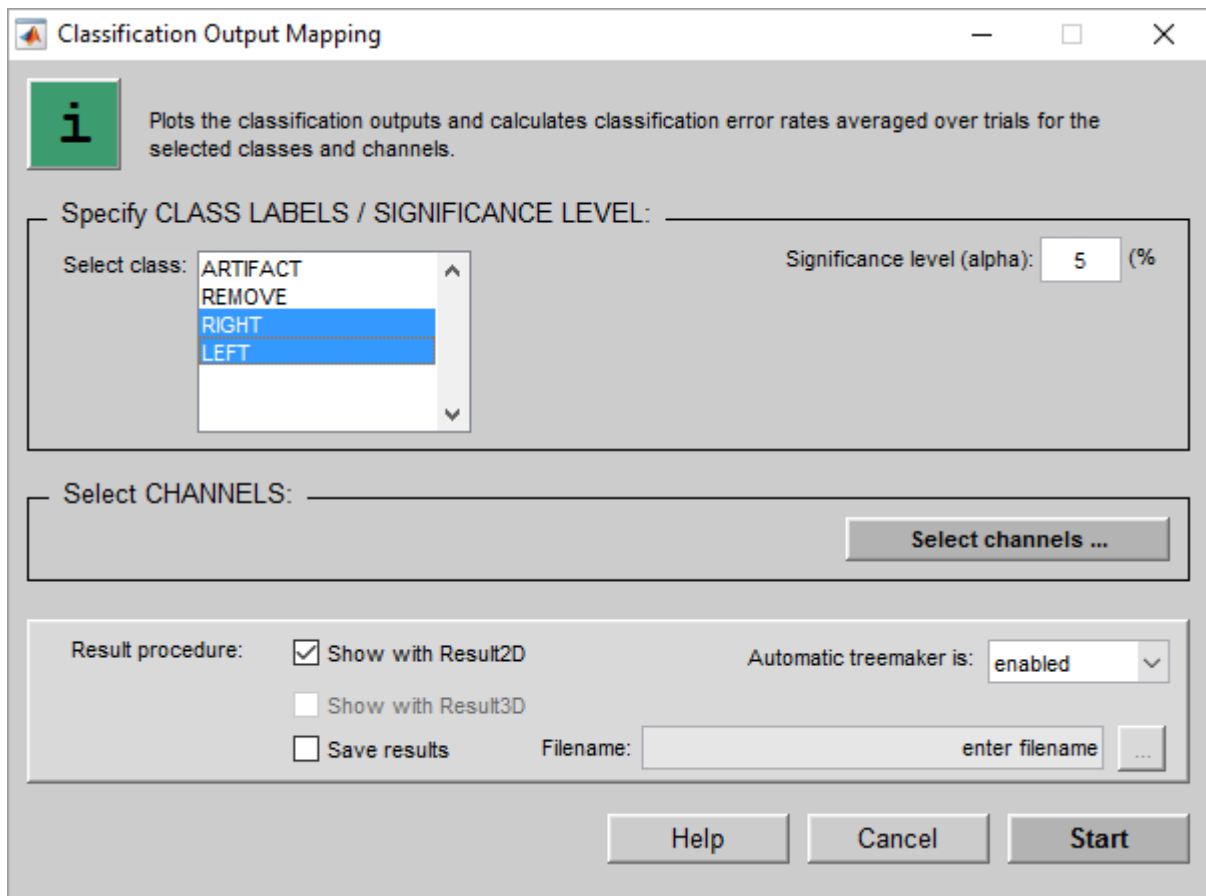
Perform the following steps:

1. Load the data-set `mi_2class_80trials.mat` from

`Documents\gtec\gBSanalyze\testdata\Classify\classifieddata`

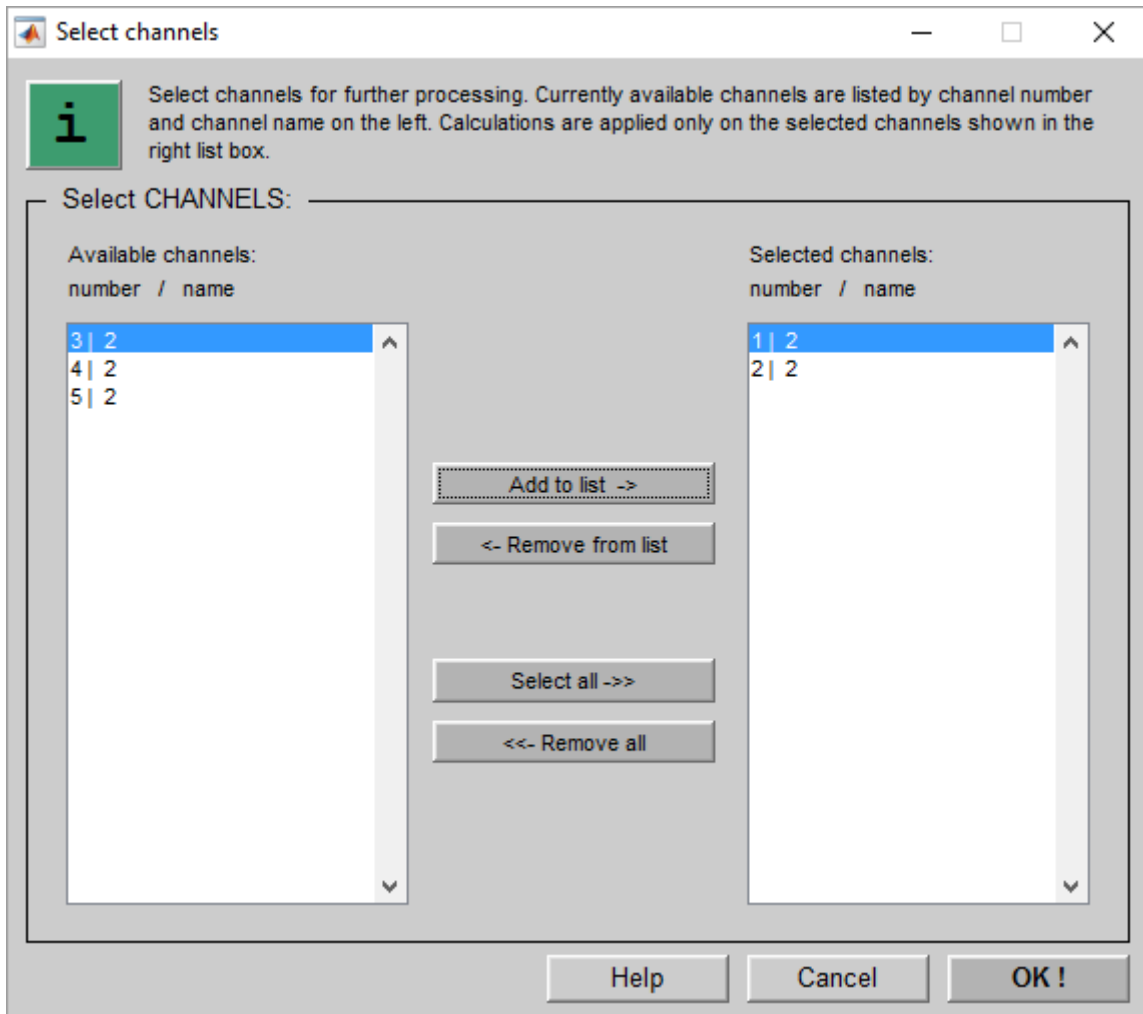
into the Data Editor. The Data Editor displays 5 channels with classification results from a two-class Motor Imagery BCI experiment, which was created with linear discriminant analysis (LDA). The first two channels display the two outputs of the LDA classification, channels 3 and 4 represent the probabilities that the data would be assigned to one of the two classes and channel 5 indicates the number of this class.

2. Open the **Classification Output Mapping** window from the **Classification** menu, select the **RIGHT** and **LEFT** classes in the **Select Class** listbox and select a value of 5 % for the **Significance level (alpha)** parameter.



3. Open the **Select channels** menu by clicking on the **Select channels** button. Then, add (into the **Selected channels** list) the first 2 channels displayed into the **Available channels** list, then press the **OK** button. The first channel is assigned to the first

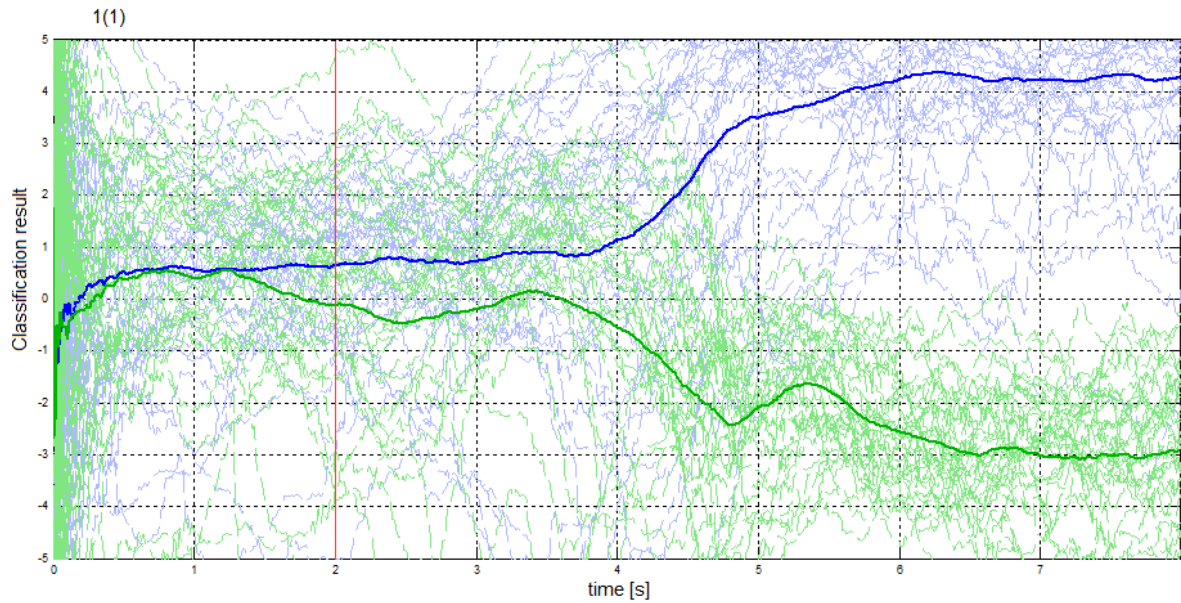
selected class (RIGHT), and the second channel is assigned to the second selected class (LEFT).



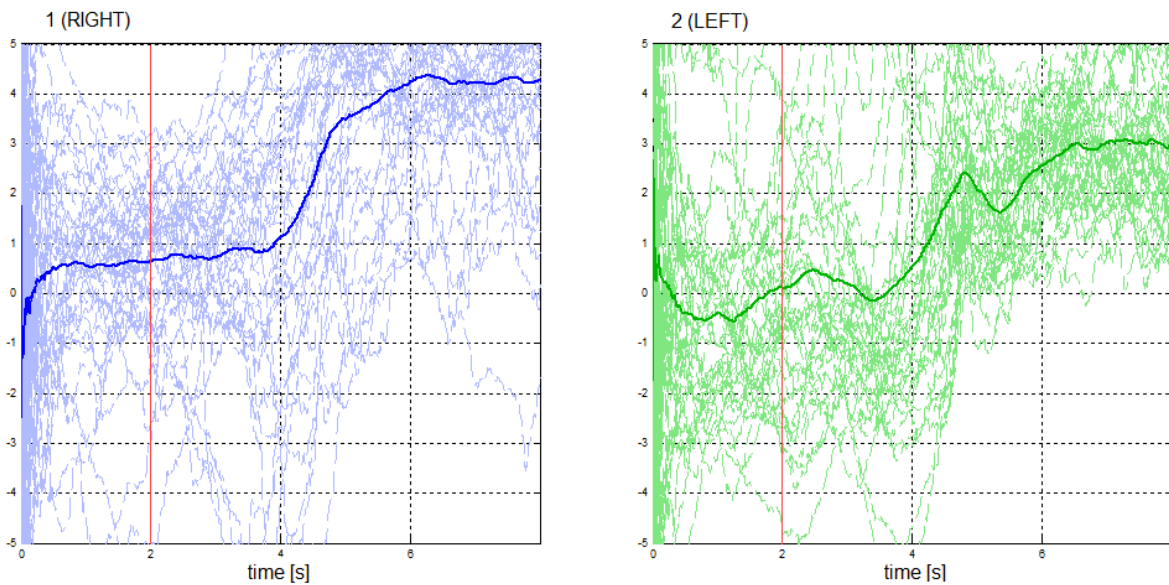
4. Check the **Show with Result2D** and **Save results** boxes, enter a filename to store the results and press the **Start** button.

ON the first page, gResult2D displays the trials (dashed lines) and trial averages (solid lines) for both classes. The assigned colors are blue for the first selected class (RIGHT) and green for the second selected class (LEFT). The red vertical line represents the time-point of the trigger. The y-axis shows the classification result (in this case the LDA distance), and the x-axis presents the timing of a single trial in seconds.

Please note: For an easier graphical comparison between the two classes, the second output channel (assigned to class LEFT) was inverted.

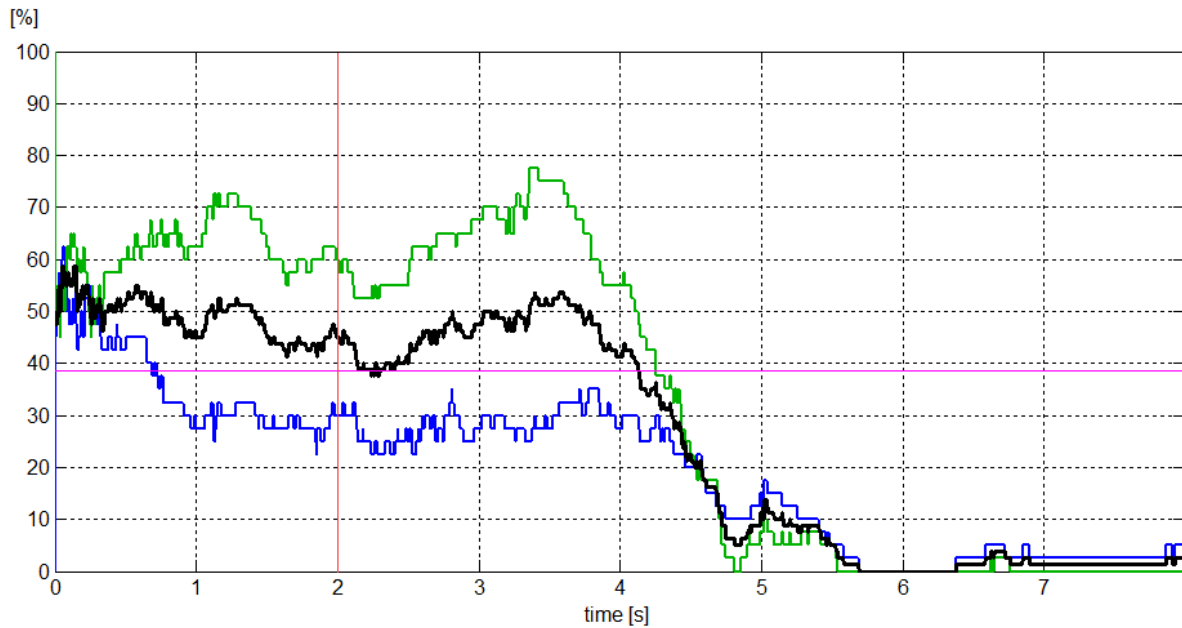


The second gResult2D page displays the trials and trials averages for each class in separate plots, as in the example below. This time the second output channel was not inverted.



The third gResult2D page displays the classification error time courses for individual classes (blue for class RIGHT and green for class LEFT) and the total error (black), averaged over all trials.






The violet horizontal line represents the upper border of the confidence interval. Based on it, the user can decide if the achieved error rates are statistically significant.

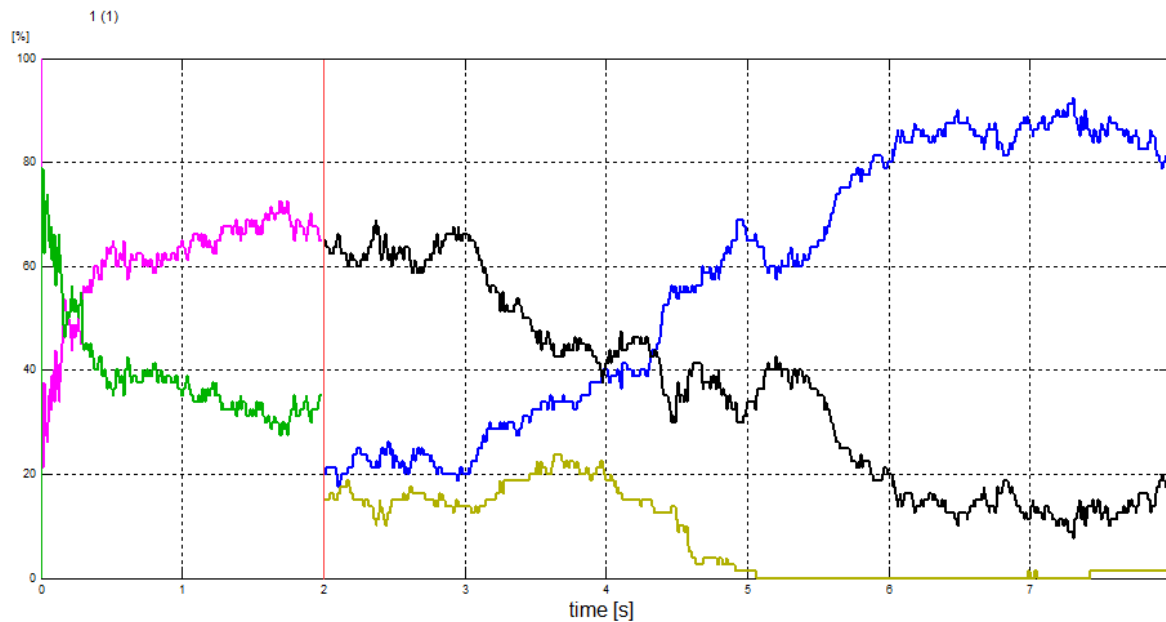


5. Repeat steps 1 to 4 but select channels 3 and 4 to map the classification probability outputs. The gResult2D will contain the same pages described before.
6. Repeat steps 1 to 4, but select channel 5 only.

If the **Apply Classifier** was done with the **Add zero class** parameter checked, the first two pages of gResult2D will contain the same features as presented before, page number 3 will display the statistical measures of the classification process, and page number 4 will contain the classification error rates.

The color assignment for the statistical features shown in the third page of gResult2D is presented below:

	True True Positive (TTP) – correct detections after the cue.
	False True Positive (FTP) – false detections after the cue.
	False Negative detections (FN) – all zero-class samples after the cue.
	False Positive detections (FP) – all detections before the cue that are not assigned to zero-class.
	True Negative detections (TN) – all samples correctly assigned to zero-class before the cue.



To perform the example shown above from the MATLAB command line, please use the following code:

%Load Data

```
P_C=data;
File=['C:\Users\' getenv('USERNAME') '\Documents\gtec\gBSanalyze' ...
'\testdata\Classify\classifieddata\mi_2class_80trials.mat'];
P_C=load(P_C,File);
```

%Classification Output Mapping

```
ClassIndex=[3 4];
ChannelExclude=[3 4 5];
TrialExclude=[];
FileName=['C:\Users\' getenv('USERNAME') '\Documents\gtec\gBSanalyze\'...
'testdata\Classify\Classifieddata\classificationoutputmapping.mat'];
SignificanceLevel=[5];
ProgressBarFlag=1;
V_O = gBSclassificationoutputmapping(P_C,ClassIndex,ChannelExclude,...
TrialExclude,FileName,SignificanceLevel,ProgressBarFlag);
result2D = CreateResult2D(V_O);
gResult2d(result2D);
```

Data Access

g.BSanalyze stores the feature matrix and the classifier data in specific objects. To access both objects the get and set commands can be used:

Using the get Command

The get method provides a way to access the object entries

Syntax

```
get(C_O_S, 'PropertyName')
```

returns the value of the property *'PropertyName'* of the object C_O_S

Example

```
get(C_O_S, 'out_err')
```

Using the set Command

The set method provides a way to set object properties.

Syntax

```
set(F_O_S, 'PropertyName', 'PropertyValue')
```

assigns the *'PropertyValue'* to the specified *'PropertyName'* of the object.

Example

```
set(F_O_S, 'SamplingFrequency', 128)
```

Accessing the Feature Matrix

g.BSanalyze stores feature matrix data in an object called F_O of class featurematrix. If the data are stored to harddisk the name of the data object is changed to F_O_S.

Entry	Description
<i>Features</i>	Feature matrix of each time point
<i>ClassLabels</i>	Class label for each sample. Each row corresponds to a class
<i>Interval</i>	Mode 1: [Start Step End] Mode 2: [Time Point 1 - Time Point 2 ...] Mode 3 : [Segment 1 Start - Segment 1 End Segment 2 Start - Segment 2 End ...]
<i>Classes</i>	Class names
<i>ChannelExclude</i>	Number of channels which were excluded
<i>FileName</i>	Name of featurematrix file
<i>SamplingFrequency</i>	Sampling rate of data file
<i>Mode</i>	Mode=1 ... generated from trial attributes Mode=2 ... generated from time points Mode=3 ... generated from segments
<i>TrialNumber</i>	TrialNumber for each feature matrix trial

Accessing the Classifier Object

g.BSanalyze stores classifier data in an object called C_O of class classifierobj. If the data are stored to harddisk the name of the object is changed to C_O_S.

Entry	Description
<i>Out_err</i>	Stores the classification error and classification time point. First row: time point Second row: total error Third row: error of class 1 ... N-row: error of class N
<i>Out_clsfyr</i>	Classifier or weight vector for each time point. Each row corresponds to a time point.
<i>Out_erridx</i>	Trial id for wrong classified trials for each class and time point
<i>Out_clstest</i>	Classification result for each time point
<i>Out_rmse</i>	Root mean squared error (only for neural networks)
<i>TPC</i>	Trials per class
<i>Interval</i>	Mode 1: [Start Step End] Mode 2: [Time point 1 start - time point 1 end Time point 2 start – time point 2 end ...]
<i>AttrNr</i>	Class name
<i>Methods</i>	Classification method
<i>TrainTestData</i>	CV, 100:100, 100:0, 50:50, 0:100
<i>Metric</i>	Euclidian or Mahalanobis
<i>InputObj</i>	Classifier method object
<i>Features</i>	Features for each time point
<i>PlotFeatures</i>	Number of features to plot as cloud

Help

g.BSanalyze and the g.CLASSIFYtoolbox provide a printable documentation and a function help.

The printable documentation is stored under

Your MATLAB path\gtec\gBSanalyze\Help

as gCLASSIFYtoolbox.pdf. Use Acrobat Reader to view the documentation.

To view the function help type

help gBSfunctionname

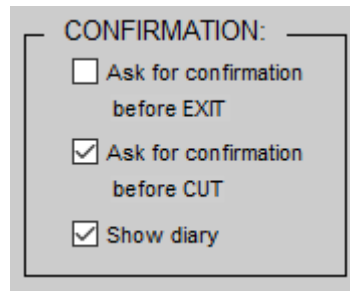
under the MATLAB command window.

To view all functions that are available in batch mode type

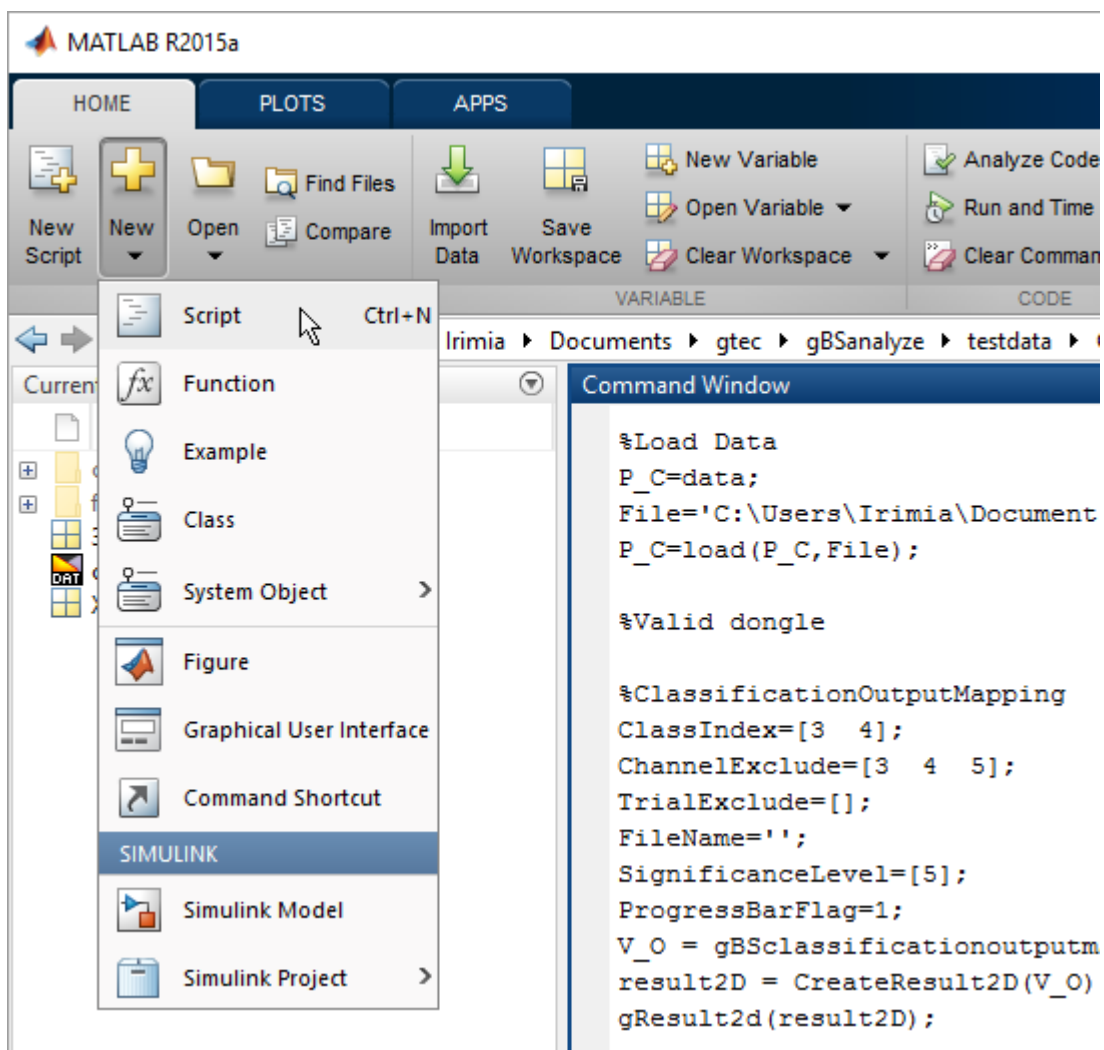
gBSfunctions

Batch Mode

The easiest way to create a batch for data processing is to perform the analysis under the Data Editor with the graphical user interfaces. Make sure that the **Show diary** checkbox is enabled in **Appearance Settings** under the **Options** menu.



This forces g.BSanalyze to report all calculations in the MATLAB command window. After finishing the analysis open a **New Script** and copy and paste all commands into the file.



```
1 %Load Data
2 P_C=data;
3 File=['C:\Users\Irimia\Documents\gtec\gBSanalyze\testdata\',...
4 'Classify\classifieddata\mi_2class_80trials.mat'];
5 P_C=load(P_C,File);
6
7 %Valid dongle
8
9 %ClassificationOutputMapping
10 ClassIndex=[3 4];
11 ChannelExclude=[3 4 5];
12 TrialExclude=[];
13 FileName='';
14 SignificanceLevel=[5];
15 ProgressBarFlag=1;
16 V_O = gBSclassificationoutputmapping(P_C,ClassIndex,ChannelExclude,...
17 TrialExclude,FileName,SignificanceLevel,ProgressBarFlag);
18 result2D = CreateResult2D(V_O);
19 gResult2d(result2D);
```

Save the batch in your own directory as `mybatch.m` and start the batch under the MATLAB command window with

```
mybatch
```

For further data-sets just replace the input data file to perform the same analysis.

Product Page

Please visit our homepage www.gtec.at for

- Update announcements
- Downloads
- Troubleshooting
- Additional demonstrations



contact information

g.tec medical engineering GmbH
Sierningstrasse 14
4521 Schiedlberg
Austria

tel. +43 7251 22240
fax. +43 7251 22240 39
web: www.gtec.at
e-mail: office@gtec.at