Exercise 2: "Attention BCI"

Introduction

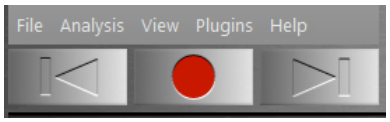
Multi-voxel pattern classification (MVPC) allows to detect differences between conditions with higher sensitivity than conventional univariate analysis by focusing on the analysis and comparison of patterns of activity. In such a multivariate approach, data from individual voxels within a region are jointly analyzed. Besides statistical multivariate tools (e.g. MANOVA), machine learning approaches such as support vector machines (SVMs) have become very popular since they focus explicitly on generalization performance (e.g. in "brain reading" applications). In this exercise, we will train a classifier (support vector machine, SVM) on the first run of a "covert attention" experiment and test the classifier performance on the subsequent runs 2 -4. If the classifier works well, we are able to predict the current focus of attention ("local" vs "global") of the participant on a trial-by-trial basis.

Prepare/load a TBV file for the classifier training run

1. Start Turbo-BrainVoyager.
2. Open a prepared TBV file located in the "TBV_SVMClassifier" folder (ATTBCI_01_trainig.tbv). Skip the points in this section.
3. Optional (3-6): If you want to create the TBV file yourself, open a TBV file from a previous exercise (e.g. from the "Faces / Houses" experiment. Then change the name of the study (Title:) on the first page and use the "Save as" button to store the new .tbv file in the folder of the "Attention BCI" classifier data (folder "TBV_SVMClassifier").
4. Reopen the "TBV Settings" dialog and set the "WatchFolder:" to the directory containing the data from the first run of the classifier experiment (folder "ATTBCI_Analyze/Ser0001").
5. Adjust the values in the "Data Format" tab. The "DataTyp:" entries need not be changed if you start with the TBV file from the previous exercise; otherwise select the "ANALYZE_NW " format. Use the following information to fill in the values for the number of slices (22), slice thickness (3), gap thickness (0.99), field of view (192, 192) and the matrix size (64, 64). Set appropriate values in the "NrOfRows:" and "NrOfColumns:" fields (5, 5).
6. Now switch to the "Time Course" tab. You may select a provided protocol (.prt) file using the "Browse" button. Set the correct value in the "NrOfVolumes:" entry (405) and for the "NrOfVolumesToSkip" entry (4). Set the "TR:" value to "1250". The "ProjectName:" and "SlicePrefix:" names are only relevant for saving the data after real-time analysis has been completed; enter, for example, "ATTBCI_01_trainig" in the "ProjectName:" field and keep the default setting performed for the "SlicePrefix:" field.

Performing real-time analysis

1. To start the prepared real-time analysis, click the red "Record / Stop" button (see below):



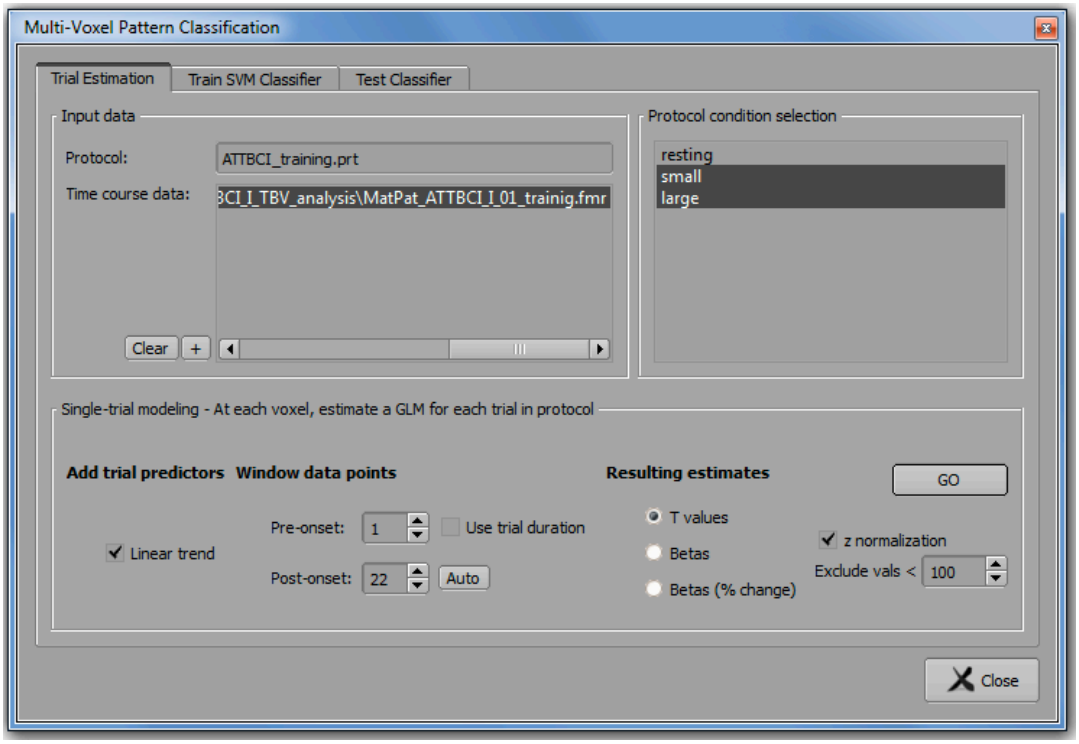
2. Inspect different regions: Do you find active (sub-)regions that discriminate between attention to the small vs attention to the large rectangels?
3. Change the smoothing parameters in the "TBV Settings" dialog, e.g. to a kernel of 4 and then 8 mm. Does this help to find regions with differential activity?
4. Create a large ROI that will be used to train the classifier. For this purpose, increase the "MultiSliceROI" option from 1 slice to about 20 slices. This will help to select a large region covering most of the visual cortex.
5. Save the defined ROI to disk - it will be used for classifier training and prediction.

Training of the SVM classifier using data of first run

1. Start Turbo-BrainVoyager.
2. If not automatically selected, choose the TBV file for the first run that you have loaded or prepared in the previous step.
3. Click the "Reload Data" button to quickly re-load the analyzed data or re-run the real-time analysis and load the previously stored ROI file (e.g. "ATTBCI_01_trainig.roi"). The screen should then look similar as shown below:

4. Now invoke the "Multi-Voxel Pattern Classification" dialog by clicking the "SVM Training" item in the "Analysis" menu. In the "Input data" field of the dialog (see below) you can add all runs that you want to use to train a classifier. In our case we have at present just one run (the first one) that we will add by clicking the "+" button. The protocol linked to the selected functional run is shown in the "Protocol" field and the conditions defined in that protocol are listed on the right side in the "Protocol condition selection" box. Keep the automatic selection of the two conditions "small" and "large" serving as labels for two classes. The selection of two

condition names is the minimum needed to train a classifier. You can, however, select more conditions if available; do not include, however the baseline (here "resting") condition. In order to select/deselect a condition, click on a condition while holding down the Ctrl (Windows/ Linux) or Command (Mac) key.



5. One major task of machine learning tools is to be able to learn from examples and to transfer the gained knowledge to new data. A classifier, for example, learns to associate presented "exemplars" with a "label". Exemplars in the fMRI context are activity patterns across a set of voxels and the "class labels" are the different conditions of the performed experiment. In order to provide a set of exemplars to a classifier, responses of each trial must be extracted/estimated for each voxel. For a voxel, the response to one trial consists usually of a single value, which may be obtained, e.g. by estimating the amplitude of that trial's BOLD response using single-trial GLMs.

6. The "Single-trial modeling" field in the lower part of the dialog will use the specified functional input data to run a General Linear Model for each trial of the specified conditions at each voxel. This requires specifying a temporal window around the onset of each trial. The easiest way to specify an interval long enough to capture the expected trial's BOLD response is to set a "Pre-onset" time (set to "1") and a "Post-onset" time (set to "22"). The "Auto" button can be used to select a useful post-onset value.

7. As default, TBV builds a design matrix for the single-trial data based on a fixed standard two-gamma HRF function. It is also possible to add a linear trend predictor to remove a linear signal drift in the trial window. Include this confound predictor by checking the "Linear trend" option.

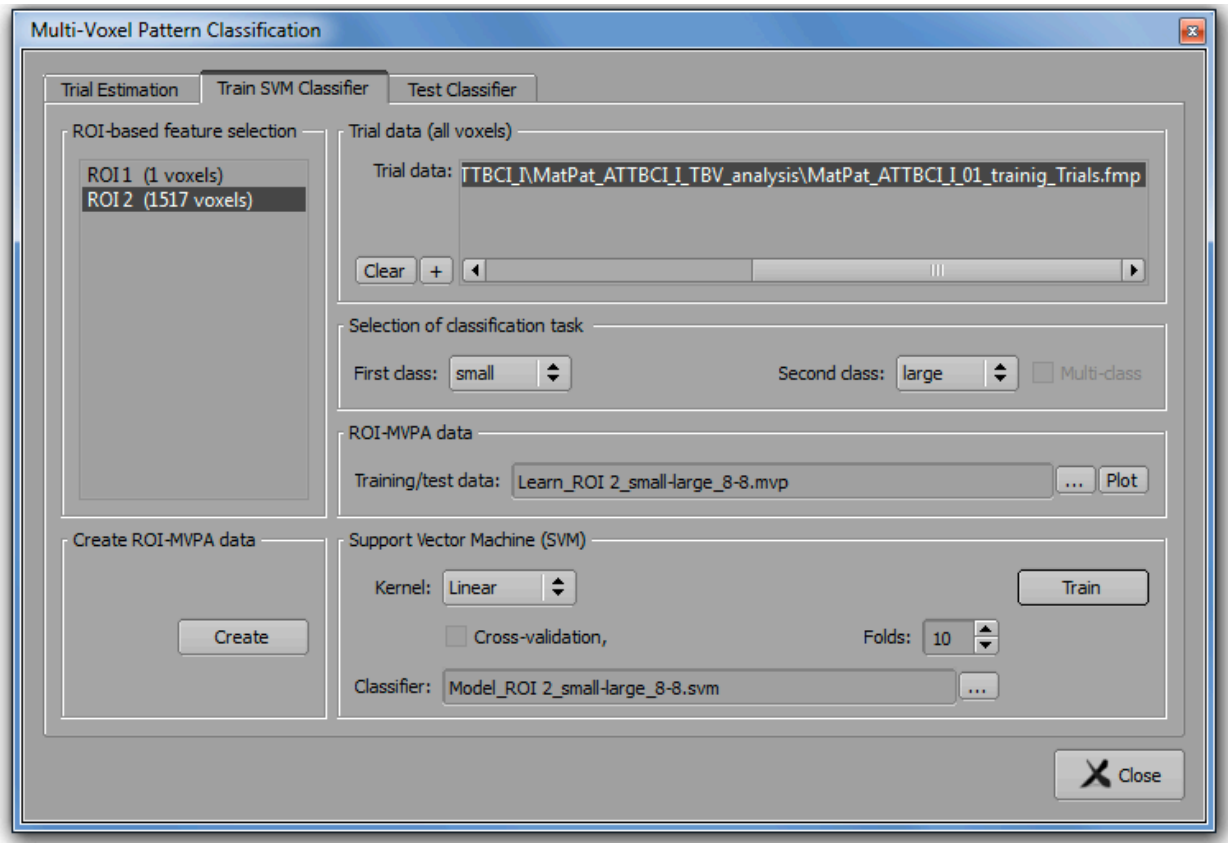
8. From the single-trial GLM, the value of the main predictor that estimates the amplitude of the BOLD response is stored for further analysis for each trial at each voxel. The estimated amplitude value might be stored as a raw beta value, as a percent change beta value (relative to baseline) or as a t value. We keep the default "T values" option since t values are usually more robust (since they are normalized to the variability of the estimate) than the beta values. Furthermore, the data in the window can be z-normalized prior to applying the GLM. Keep the "z-normalization" option turned on. With z-normalization, also beta values would provide

stable estimates - you can check the results using (percent change) beta estimates as an optional exercise.

9. To start the trial estimation procedure using the specified options, click the "GO" button (see snapshot above). The estimated t values (or beta values, if selected) will be stored in a set of FMP files, one for each source FMR file; in our example, only one FMP file (ending with "_Trials.fmp") will be created.

10. Using the estimated single-trial data, we can now train a classifier to learn to discriminate the distributed activity patterns evoked by the two different attention conditions. In order to test generalization performance for new data, we will use runs 2-4 subsequently.

11. A classifier is usually not applied to the whole data but to a restricted part of the brain ("feature selection" step). Using a region-of-interest (ROI), we can extract training and testing data for any subset of voxels. In the "Multi-Voxel Pattern Classification" dialog, switch to the "Train SVM Classifier" tab (see snapshot below). In the "Trial data" box, you will find the produced FMP file created in the previous step.



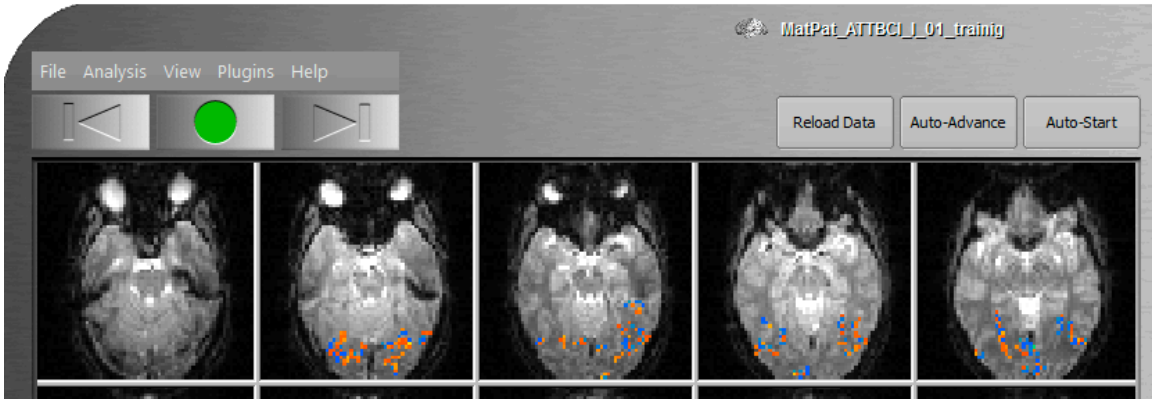
12. The "Selection of classification task" field shows the two condition labels ("small" for the first class and "large" for the second class). The "Multi-class" option is disabled since we use only two classes. Keep the default class selections.

13. The "ROI-based feature selection" field shows the ROIs currently defined. Select the big ROI ("ROI 2") containing 1517 voxels in the visual cortex. To extract the beta values from the FMP files for those voxels in the selected ROI, click the "Create" button in the "Create ROI-MVPA data" field. As a result of the launched process the file "Learn_ROI 2_small-large_8-8.vmp" have been created containing the multi-voxel patterns (".mvp" file extension) for the voxels of the selected ROI. Note that the file name reveals the purpose of the file ("Learn"), the region from which the trial data has been extracted ("ROI 2"), the classes/conditions used ("small-large"), and the number of exemplars/trials per class (8-8).

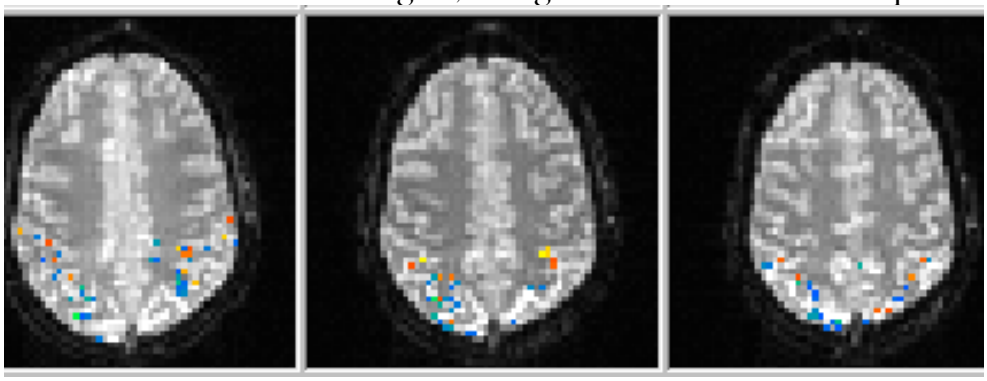
14. You may visualize the training patterns by clicking the "Plot" button in the "ROI-MVPA data" field. The displayed matrix corresponds to the main content of a MVP file, which is a matrix with a voxel dimension (x-axis in the MVP plots) and a dimension with trial response estimates (y-axis in the MVP plots).

15. The "Support Vector Machine (SVM)" field can be used to specify a classifier for training. In the "Kernel" combo box, a linear or non-linear SVM can be specified. We will keep the default setting "Linear", which is important since we want to visualize the trained "weights" for each voxel and this requires a linear classifier for proper interpretation. It is also possible to optimize parameters of a classifier using a "cross-validation" technique, which we will not use in this exercise (for details, check the User's Guide). Click the "Train" button to start the learning process. You will see in the "Classifier" text box of the "Support Vector Machine (SVM)" field the name of the trained classifier (e.g. "Model_ROI 2_small-large_8-8.svm"). Besides a different extension (".svm"), the name is identical to the name of the training data except that the sub string "Learn_" has been replaced with "Model_".
Note. The "Log Pane" in the "Test Classifier" tab will show the result when "predicting" a class ("1" --> "small", "2" --> "large") from the activity pattern of each trial.

16. After clicking the "Train" button the result of training - a weight value for each voxel of the used ROI is visualized in the "Brain Window" view as a map (see below).



17. To see the most relevant weights, change the threshold of the map to a value around "3.0".

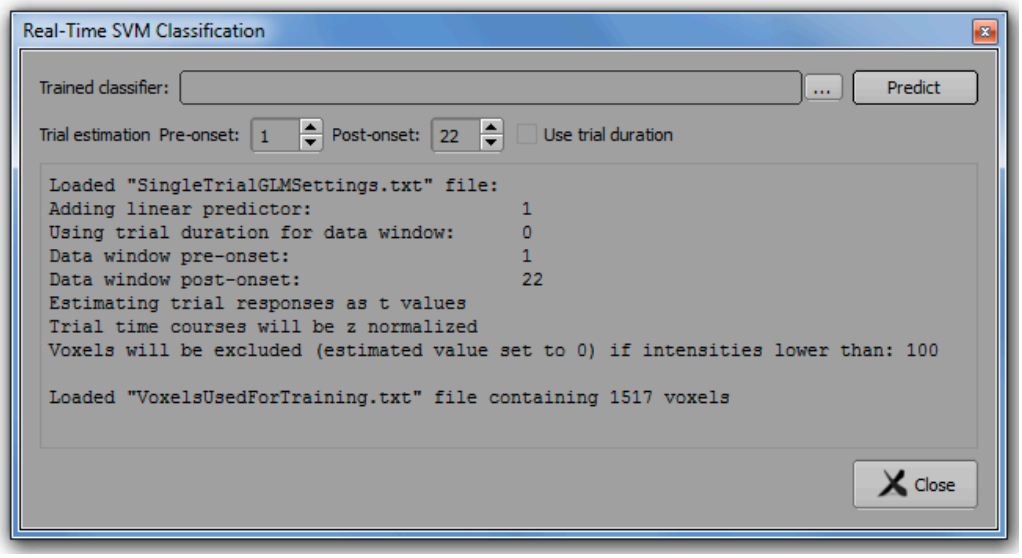


The snapshot above shows large weights in the parietal lobe, a region involved in attention processing.

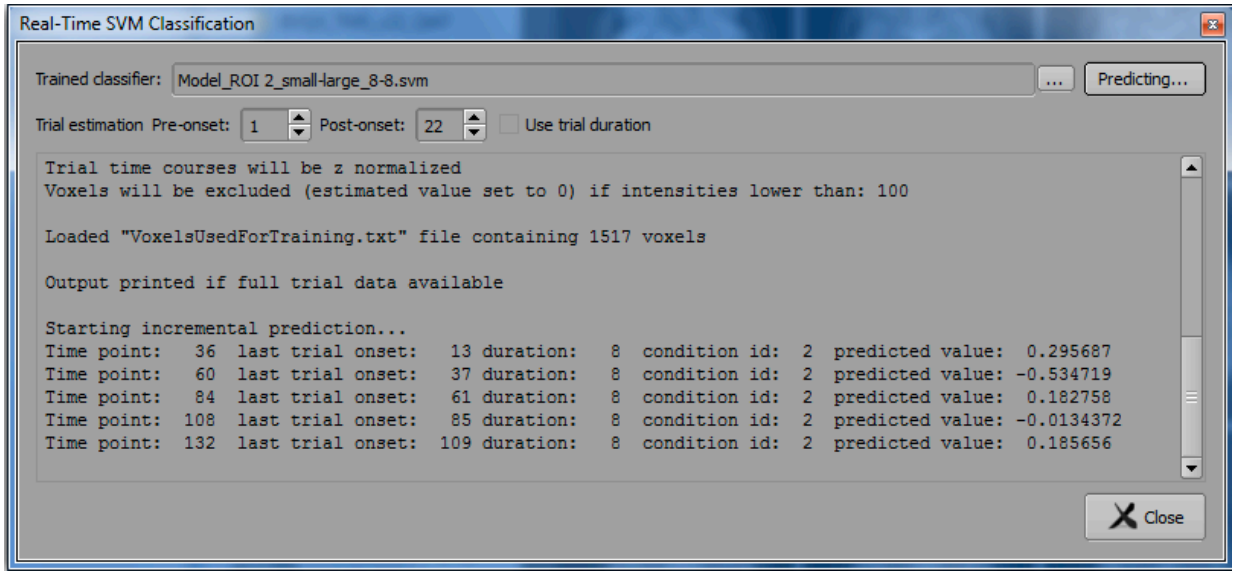
Note: The "Test Classifier" tab can be used to test the performance of the trained classifier with the data of a whole run as input. A more interesting prediction test is described in the next paragraph.

Testing the trained SVM classifier on a trial-by-trial basis

1. Close the "Multi-Voxel Pattern Classification" dialog.
2. Create or load the .TBV settings file for the first test run (run 2). Make sure that the "WatchFolder:" now points to the second run, i.e. it must be set to the "Ser0002" folder.
3. Open the "Real-Time SVM Classification" dialog (see below) by clicking on the "Real-Time SVM Classification" entry in the "Analysis" menu.



4. The dialog loads the settings used to specify trial estimates from a stored "SingleTrialGlMSettings.txt" file. It also loads the selected ROI during pattern creation from a stored "VoxelsUsedForTraining.txt". Besides this automatically loaded information, the trained classifier needs to be selected using the browse ("...") button next to the "Trained classifier" text field; select the classifier file (e.g. "Model_ROI 2_small-large_8-8.svm") created in the previous task.
5. Start real-time processing of the first test run. To start the incremental classifier, click the "Predict" button in the "Real-Time SVM Classification" dialog. As soon as enough data points become available, TBV will make a prediction to which class the current activity pattern belongs. The snapshot below shows the state after five trials have been predicted. Each line after the "Starting incremental prediction..." text shows data of one trial. The line contains information about the time point when the prediction has been performed, the trial onset, duration, condition id and finally the predicted value. Note that the condition id is usually uninformative in true real-time scenarios since each trial gets the same condition label (here "2"); in order to test the performance of a classifier, one might also code conditions executed by the subject in that trial (if known, i.e. specified by the experimenter).



If the predicted value is larger than 0.0, class 1 (here "small") is predicted; if the predicted value is smaller than 0.0, class 2 (here "large") is predicted. Since in our experiment the subject is instructed to change attention from large to small to large to small and so on, all shown 5 trials are correctly predicted in this example.

6. How does the classifier work? As soon as all data points are available to estimate a trial response using a GLM, response values (t, beta or percent change) for all ROI voxels are calculated and submitted as a test pattern to the classifier. Information about the processed trial as well as the output value predicted by the classifier is printed within one line in the Log pane. The first line above, for example, indicates that the first trial is processed at time point 36 since at this point the data window for the trial that started at time point 13 is complete (with respect to the used trial duration and post-trial settings). The printed line also shows the duration of that trial (8 time points) and the id of the corresponding condition in the protocol (2). The final value is the predicted class label.

7. Quantify the performance of the classifier based on the number of correct classifications related to the overall number of predictions.

8. Run the same analysis also for test runs 2 and 3. Do you get similar results? If not, can you explain why?