Neuromethods (2020) 151: 129–156
DOI 10.1007/7657_2019_26
© Springer Science+Business Media, LLC 2019
Published online: 12 September 2019

Multivariate Methods to Track the Spatiotemporal Profile of Feature-Based Attentional Selection Using EEG

Johannes Jacobus Fahrenfort

Abstract

This chapter provides a tutorial style guide to analyzing electroencephalogram (EEG) data contingent on feature-based attentional selection. It is targeted at researchers that currently investigate attentional processes using univariate methods but consider moving to multivariate analyses. The chapter starts by providing examples of classical univariate analysis, in which the EEG signal occurring ipsilateral to the target is subtracted from the signal that occurs in a contralateral electrode (i.e., the classical N2pc, an interhemispheric posterior negativity emerging around 180–200 ms). Next, it shows how the same type of information can also be identified using multivariate pattern analysis (MVPA). MVPA does not restrict one to contrast attentional selection in opposite hemifields but also allows one to assess attentional selection on the vertical meridian, or even within a quadrant of the visual field, opening up new avenues for research. The chapter demonstrates how to visualize topographic maps of attentional selection when using MVPA and shows how to assess timing onsets using the percent-amplitude latency method. Finally, it shows how a forward encoding model enables one to characterize the relationship between a continuous experimental variable (such as attended targets positioned on a circle) and EEG activity. This allows one to construct brain patterns for positions in the visual field that were never attended in the data that was used to create the forward model. This chapter is intended as a practical guide, explaining the methods and providing the scripts that can be used to generate the figures in-line, thus providing a step-by-step cookbook for analyzing neural time series data in the field of feature-based attentional selection.

Keywords Feature-based attention, Attentional selection, EEG, Univariate analysis, N2pc, MVPA, Multivariate pattern analysis, Classification, Decoding, BDM, Forward encoding model, Inverted encoding model, FEM

1 Introduction

Human observers are very good at extracting information from the visual field based on some relevant feature dimension, such as color. For example, when asked to determine whether the red element in a search display is a digit or a letter, they can do so very quickly and with very high accuracy (see Fig. 1). This ability is often referred to as feature-based attentional selection. There is a long history of investigating feature-based attentional selection (from here on referred to as attentional selection) using EEG [1, 2]. Traditionally, attentional selection is investigated in EEG using univariate analyses, in which targets are presented in one of two hemifields [3]. Observers detect

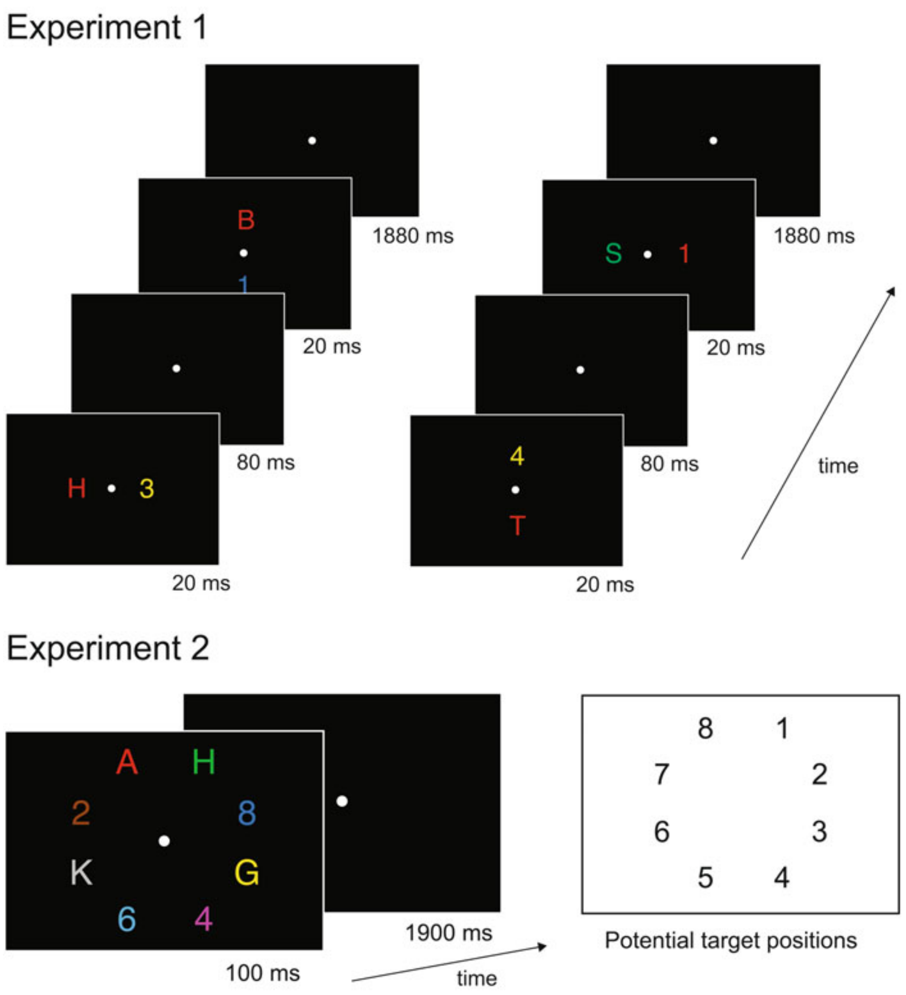

Fig. 1 Task structure in Experiment 1 and Experiment 2. *Top*: Example trial time lines of Experiment 1. There were two types of trials: the first display contained items on the horizontal meridian and the second display items on the vertical meridian, or the first display contained items on the vertical meridian and the second display items on the horizontal meridian. Subjects were required to determine whether a color-defined target was a digit or letter. The target color remained constant within a session (red in this example). Potential target colors were red, green, blue, or yellow (counterbalanced across subjects). Within blocks, subjects either had to detect a target in the first display (D1 blocks) or they had to detect a target in the second display (D2 blocks). We only analyzed task relevant displays. *Bottom*: Task and conditions in Experiment 2. *Bottom left panel*: Trial time line of Experiment 2. Subjects were asked to determine the identity of a colored target (letter or digit); target color could be red, green, or blue (counterbalanced across subjects). *Bottom right panel*: Positions used in Experiment 2, counted clockwise, one position contained the target, the other positions were occupied by distractors

a feature-defined target either on the left or on the right of fixation, and an electrophysiological marker of this selection process is identified by subtracting the event-related potentials (ERPs) on the ipsilateral side of the target from the ERPs occurring on the contralateral side of the target, typically using electrodes PO7 and PO8. This process results in an EEG component that is referred to as the N2pc.

Recently, we have shown how such a signal can also be extracted using multivariate pattern analysis (MVPA) [4]. This chapter compares the traditional method of obtaining electrophysiological markers of attentional selection to those that are obtained through MVPA. First, it explains how to compute the N2pc and provides examples of the N2pc. Next, it compares these to a classification approach in which a neural signature of attentional selection is generated by exploiting the multivariate nature of the EEG signal. Although some aspects of multivariate classification are explained, the focus of the chapter is on illustrating how to execute this analysis oneself. For details regarding multivariate classification, I refer the reader to more specialist texts [5, 6]. The chapter also shows how to properly extract and compare temporal onsets, as well as how to plot topographic maps based on forward-transformed classifier weights.

Finally, the chapter explains how a signature of attentional selection can also be captured using a forward encoding model and how this model can be used to construct cortical activity for conditions that did not occur in the experiment. Potential caveats of the forward encoding approach are briefly discussed. More detailed information is provided with respect to the forward encoding approach than is done for classification, although here too the focus is on practical application. Together, this provides a practical manual for how univariate and multivariate analyses can be carried out in the field of attentional selection, supplying both scripts and some theoretical background along with the analyses. Because of the focus on applicability, it does not follow a standard introduction-methods-results-discussion structure but rather a structure that allows the reader to understand the procedures by reading the explanations along with the results and the scripts that produce these results. Hopefully, this helps the reader to more easily reproduce and apply these analyses to their own data.

The analyses that are presented in this chapter were performed in MATLAB (MathWorks Inc., USA) using the freely available Amsterdam Decoding and Modelling (ADAM) toolbox v1.08-beta [5]. The reader can reproduce the analyses in this chapter in MATLAB by installing the ADAM toolbox and its dependencies from https://github.com/fahrenfort/ADAM/tree/1.08-beta (follow the instructions under "install") and downloading the data and/or results from https://osf.io/r67sc/. To reproduce the results, the reader can either choose to download the preprocessed data from that location (EEGLAB_DATA.zip) and compute the single subject first-level results from scratch or download the first-level results (MVPA_RESULTS.zip) and compute only the group results. Preprocessing of EEG data is not treated or explained in this chapter, as it is not sufficiently relevant to understand the methods that are presented here. As a general remark, I recommend to keep preprocessing to a minimum. The only preprocessing that was applied to these data was (1) epoching the continuous data into trials, (2) performing independent component analysis to remove components reflecting eyeblinks, and (3) removing trials that contain horizontal or vertical eye movements based on the electrooculogram (EOG). No offline high-pass filtering or low-pass filtering was applied to the data, as this can produce unwanted shifts in temporal onsets and produce spurious results; e.g., see [7, 8]. If desired, the preprocessing script that was applied to the data (as well as all the other scripts) can be downloaded from the Open Science Framework (OSF) at the previously noted location (SCRIPTS.zip).

Most of the analyses that are presented in this chapter have also been published in a *Scientific Reports* article that is freely available in the public domain; see [4]. Experimental details that are not reported in this chapter are deemed irrelevant to understanding the methods presented here but can be found in that publication if needed. Further note that small differences between the results from that publication and the results as presented in this chapter are due to the fact that the data in this chapter were analyzed without applying a high-pass filter.

2 Characterizing Attentional Selection Using the N2pc and Multivariate Classification (MVPA)

In two separate experiments, subjects performed the tasks in Fig. 1 (top, Experiment 1, $N = 12$; bottom, Experiment 2, $N = 15$) while EEG was collected. In both experiments, subjects were instructed to fixate a dot in the middle of the screen while reporting the category of a target color (digit or letter) by pressing a response key. In both experiments, the target color was the same throughout the experiment for any given subject, while target colors were counterbalanced across subjects.

In Experiment 1, each trial contained two successively presented stimulus displays. Each display contained two items on opposite sides, one in the target color (e.g., red in Fig. 1) and another one in a nontarget color. In different blocks of the experiment, subjects either had to report the target in the first display or report the target in the second display. One of the two displays on each trial contained a target-nontarget color pair on the horizontal midline (to the left and right of fixation), and the other display contained a target-nontarget color pair on the vertical midline (above and below of fixation). Only task-relevant displays were analyzed. Presentation sequence (vertical stimulus pair preceded by horizontal pair or vice versa) was randomized across trials. The goal of the experiment was to identify a neural signature of attentionally selecting the relevant item. In Experiment 2, each trial contained a single search display comprising eight items in a circular array. The target could appear in any of the eight locations (Fig. 1, bottom). Other than that, the task was identical. In what follows, the chapter describes how to characterize the unfolding of attentional selection using EEG signals acquired during these experiments. The data are analyzed using the ADAM toolbox [5]. At every step, an ADAM script is provided to perform a given analysis, along with a brief explanation of the parameters that are defined for that analysis. Next, the figures are presented that are produced by the analysis.

Typically, EEG data are analyzed by first computing the results at a single subject level. This is also called the first-level analysis. After obtaining first-level results, a statistical analysis is performed at the group level. The ADAM toolbox obtains the first-level results by reading single subject EEG data from disk, performing the relevant univariate and multivariate analyses on these single subject data, and subsequently saving the resulting first-level single subject results to the hard drive. Group-level results are then computed and visualized by reading in the single subject results from disk and subsequently performing and plotting group-level statistics. Below is the script to run the first-level analyses of the first experiment, which can be executed provided that the reader has installed a working copy of MATLAB, the ADAM toolbox and its dependencies, and has downloaded the preprocessed EEGLAB files from OSF. The only thing that requires setting are the proper input directory and output directories. Programming experience is not required to execute these analyses, but the reader should know how to open and execute .m script files in MATLAB and/or running snippets of code from the MATLAB Command Window.

```
%% some general settings regarding experiment 1
filenames = {     'TopDown_1__merged'  'TopDown_2__merged' 'TopDown_3__merged' ...
                  'TopDown_4__merged'  'TopDown_5__merged' 'TopDown_7__merged' ...
                  'TopDown_8__merged'  'TopDown_9__merged' 'TopDown_10__merged' ...
                  'TopDown_12__merged' 'TopDown_14__merged' 'TopDown_16__merged'};
cond_left =      [1 3 11 13]; % these condition codes specify target position
cond_right =     [2 4 12 14];
cond_top =       [5 7 15 17];
cond_bottom =    [6 8 16 18];

% first level analysis
cfg = [];
cfg.datadir = 'C:\EXP1\EEGDATA';
cfg.filenames = filenames;      % specifies the filenames
cfg.erp_baseline = [-.2,-.1];   % baseline period
cfg.resample = 250;             % lower sampling rate to save time
cfg.nfolds = 10;                % number of folds used in the k-fold
cfg.model = 'BDM';              % 'FEM' for forward encoding model
cfg.channels = 'ALL';           % channel pooling

% classify left versus right
cfg.class_spec{1} = cond_string(cond_left);
cfg.class_spec{2} = cond_string(cond_right);
cfg.outputdir = 'C:\EXP1\RESULTS\LEFTRIGHT';
adam_MVPA_firstlevel(cfg);

% classify top versus bottom
cfg.class_spec{1} = cond_string(cond_top);
cfg.class_spec{2} = cond_string(cond_bottom);
cfg.outputdir = 'C:\EXP1\RESULTS\TOPBOTTOM';
adam_MVPA_firstlevel(cfg);
```

This script performs both univariate and multivariate analysis on the single subject EEG files. It operates by specifying a number of relevant parameters as fields using the cfg variable. Cfg is shorthand for configuration, and the cfg variable is used to pass these parameters to the relevant ADAM function. The actual first-level analyses are executed by the adam_MVPA_firstlevel function (which contains the configuration variable containing the relevant parameters between brackets). It does so by first reading in the epoched EEG data of individual subjects in EEGLAB format (specified in cfg.filenames) from a location on the hard drive (specified in cfg.datadir). Next, it baselines all trials to a window of (−200, −100) (specified in cfg.erp_baseline). Note that that this baseline window was chosen so that the baseline for the second search display would never overlap with the first search display (see Fig. 1, top). The data is also down-sampled to 250 Hz (specified in cfg.resample) prior to classification to expedite the classification analysis.

Next, it performs both a univariate and multivariate analysis. The univariate analysis is executed by computing ERPs for all the electrodes in the EEG data (specified in cfg.channels), separately for targets appearing on the left or on the right of the search display as well as for targets appearing on the top and the bottom of the search display (specified in cfg.class_spec). These two analyses (left versus right and top versus bottom) are executed and saved separately, in a folder called 'LEFTRIGHT' and 'TOPBOTTOM' respectively (specified in cfg.outputdir). We will see later how one can use the ERPs that the function has produced to compute the N2pc.

In addition, the function performs a tenfold (specified in cfg.nfolds) leave-one-out cross-validated multivariate classification analysis using a linear discriminant classifier ('BDM', short for backward decoding model, specified in cfg.model), and it does this across all the EEG electrodes included in the analysis (specified in cfg.channels). As explained, the script above performs two first-level analyses: the first one classifies targets appearing on the left of the search display versus targets appearing on the right of the display, and the second one classifies targets appearing on the top versus targets appearing on the bottom of the search display. The class definitions in cfg.class_spec specify which target positions are classified in the analysis. Classes are specified using the cond_string function, which merely converts integer numerals to comma-separated strings, which is the input format that is required by the ADAM toolbox.

After the first-level analyses are performed, the single subject results are stored at the hard drive location specified in cfg.outputdir, which are later read back in when performing a group-level analysis. Therefore, it is important to specify a meaningful directory name for the location of the first-level results. The directory name should reflect which analysis was performed because (1) this directory needs to be indicated when running the group-level analysis and (2) the name of this directory will be used by ADAM to denote the analysis name in graphs. Note also that one can specify quite a few additional parameters in the cfg variable during first-level analysis. Many of these are not covered here. Without specifying these fields, the analysis is performed using default values. Detailed information about some of the parameters that can be specified when performing an analysis in ADAM, as well as the meaning of some parameters (such as cross-validation) is beyond the scope of this chapter. For more information about these parameters, type "help adam_MVPA_firstlevel" in the MATLAB Command Window. In addition, a freely available open access article is available for those that require more general information regarding MVPA using the ADAM toolbox; see [5].

Next, I describe how one can perform and visualize group-level statistical analyses on these first-level results. As explained in the introduction, the traditional method of identifying attentional selection in experiments like these is using the N2pc. The N2pc is typically computed by subtracting the ERPs on the ipsilateral side of the target from the ERPs occurring on the contralateral side of the target, using electrodes PO7 and PO8 (see the left panel of Fig. 2 for the search display the subject is looking at and the electrodes that need to be subtracted to compute the N2pc). In the first group-level analysis, I illustrate how the ADAM toolbox can extract the N2pc using the ERPs from the first-level analysis, to compute a group-level N2pc for a single hemisphere. The actual

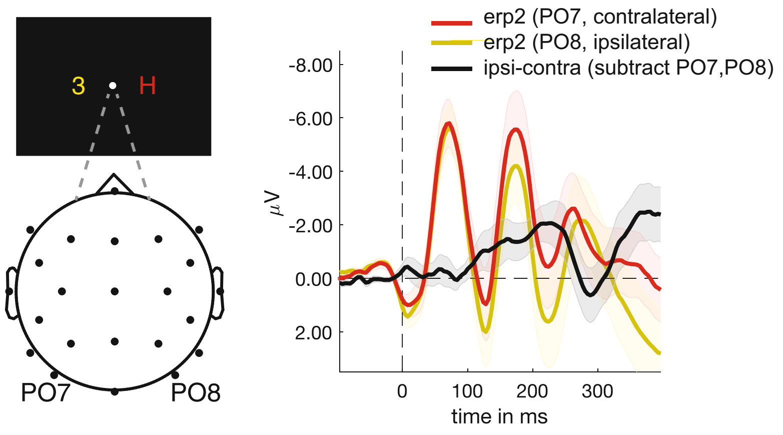

Fig. 2 Illustration of N2pc component. *Left*: Illustration of a subject attentionally selecting the red target on the right in a two-item display in Experiment 1. *Right*: ERP responses on the ipsi- and contralateral electrode to the attended item, as well as the difference between these two. This difference is the univariate N2pc, here for one attended hemifield

script to compute these ERPs and to produce the resulting group-level N2pc is given below.

```
%% extract electrodes P07 and P08, and also subtract them
cfg = [];
cfg.mpcompcor_method = 'none';
cfg.startdir = 'C:\EXP1\RESULTS';
cfg.electrode_def = {{'PO7'},{'PO8'}};
cfg.condition_def = 2;
cfg.timelim = [-100 400];
erpstats = adam_compute_group_ERP(cfg); % select the folder LEFTRIGHT when running this line
cfg.electrode_method = 'subtract';
erpstatsdif = adam_compute_group_ERP(cfg); % select the folder LEFTRIGHT once again

%% plot ERPs
cfg = [];
cfg.acclim = [-8.5 3.5];    % specifies the limits on the y-axis, not required
cfg.acctick = 2;
cfg.singleplot = 'yes';
cfg.line_colors = {[228,30,38]/255 [255,242,0]/255 [0,0,0]};
adam_plot_MVPA(cfg,erpstats,erpstatsdif);
```

The first part of this script loads the single subject data from the RESULTS folder and computes group ERPs when executing the function adam_compute_group_ERP. When this function executes, a folder selection window pops up at the location specified in cfg.startdir, after which one should manually select the LEFTRIGHT directory. Next, it extracts the single subject ERPs from that directory and computes a group-level average of these single subject results. It does so for the second class in the analysis which contained targets presented on the right (specified using cfg.condition_def) from electrode PO7 and PO8 (specified in cfg.electrode_def) within a temporal window of (100, 400) ms (specified in cfg.timelim). No statistical testing is applied (multiple comparison correction, specified as 'none' using cfg.mpcompcor_method). When the adam_compute_group_ERP function is executed, the output of the analysis is stored in a variable called erpstats. Next in the script, the analysis is performed again, now subtracting the ERP from electrode PO8 from PO7 to compute the right-hemispheric N2pc (ipsilateral minus contralateral). This is done by specifying cfg.electrode_method = 'subtract' and the running the same function again, outputting the result in the variable erpstatsdif.

The second part of the script inputs the erpstats and the erpstatsdif result variables into the adam_plot_MVPA function to plot the results. This function produces a graphical depiction of the ERPs that were computed by adam_compute_group_ERP. It plots the separate ERPs from P07 (in red) and PO8 (in yellow) as well as their difference (the N2pc, in black) together in a single figure (cfg.singleplot = 'yes'). Although not required, some additional parameters can be used to further configure the plot. For example, cfg.acclim specifies the limits of the y-axis, and cfg.acctick specifies the tick mark of the y-axis (for more information about plotting parameters, type “help adam_plot_MVPA” in the

MATLAB Command Window). Finally, cfg.line_colors specifies which Red-Green-Blue (RGB) color values to use for the consecutive plots (scaled between 0 and 1). For more information about color specifications in MATLAB type "help colormap" in the MATLAB Command Window. The result of the plotting operation is shown in the right panel of Fig. 2.

However, this only shows the N2pc for a single hemisphere. For the N2pc proper, one should compute the ipsilateral-contralateral difference separately for targets appearing in the left visual field and targets appearing the right visual field and subsequently average those subtractions. This is done using the script below.

```
%% get total N2pc
cfg = [];
cfg.mpcompcor_method = 'cluster_based';
cfg.startdir = 'C:\EXP1\RESULTS';
cfg.electrode_def = {{'PO8'},{'PO7'};{'PO7'},{'PO8'}};
cfg.electrode_method = 'subtract';
cfg.condition_def = [1,2];
cfg.condition_method = 'average';
cfg.timelim = [-100 400];
n2pcstats = adam_compute_group_ERP(cfg); % select the folder LEFTRIGHT when running this line

%% plot N2pc
cfg = [];
cfg.acclim = [-2.5 1];
cfg.singleplot = 'yes';
adam_plot_MVPA(cfg,n2pcstats);
```

The only difference with the earlier script is that this time the ERPs from both class 1 (targets appearing on the left) and from class 2 (targets on the right) are extracted (again specified in cfg.condition_def) and that for each of these, the ipsilateral electrode is subtracted from the contralateral electrode (again specified in cfg.electrode_def). The resulting subtractions are averaged (specified in cfg.condition_method) and tested against zero using a two-sided *t*-test against chance for each time sample. The statistical tests are corrected for multiple comparisons using cluster-based permutation testing (specified using cfg.mpcompcor_method). This method uses group-wise cluster-based permutation testing by taking the sum of the *t*-values for all contiguously significant time points ($p < 0.05$) and computing the number of times this sum is exceeded when computing the maximum cluster-based sum under random permutation [5, 9].

The group average that is computed by adam_compute_group_ERP is stored in variable n2pcstats and subsequently plotted using adam_plot_MVPA. The resulting figure can be found in Fig. 3, left panel. This is the "traditional" N2pc that is often reported in the literature [1, 2]. Note that the figure also contains a vertical dotted line halfway the first peak. This is a measure of the onset of the N2pc component. Note that taking the onset of the cluster itself is an unreliable way of determining the onset latency of an effect [10]. Further note that taking the peak latency is easily

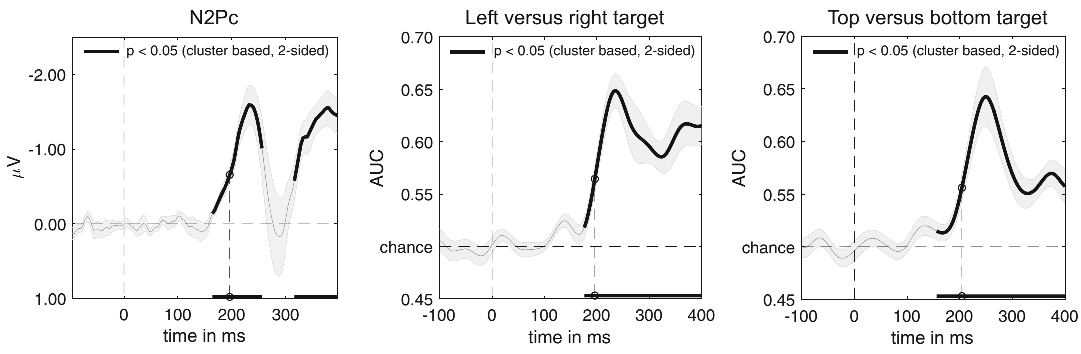

Fig. 3 Average N2pc and classification performance. *Left*: Average N2pc for left and right targets in Experiment 1. Computed using PO7 and PO8. See main text for details. *Middle and right*: Classification performance of target position for left versus right targets (middle) and classification performance for top versus bottom targets (right). Thick black lines reflect statistical tests that survive cluster-based permutation testing at $p < 0.05$. Shaded areas show ±s.e.m. Onset latency of the 50% amplitude of the peak is indicated by a vertical dotted line. Note the similar temporal evolution between N2pc and classification performance

distorted by neuronal and measurement noise [11, 12]. Instead, a relatively straightforward and reliable way of characterizing temporal onsets is to measure the onset latency as the time when the rising effect of the component has reached 50% of its full amplitude. This is the standard method implemented in the ADAM toolbox, based on freely available code [13]. The onset latency for the N2pc that is estimated this way is stored in the group stats variable, in a field called latencies. Thus, one can access the latency of the N2pc by typing n2pcstats.latencies in the MATLAB command window. The field GA (short for Grand Average) tells us what the onset latency is of the N2pc when computed this way, which for this N2pc is 196 ms. Further below, we will assess whether multivariate measures of attentional selection result in similar onset latencies.

Although the N2pc has been a very successful measure of attentional selection, it also has some prominent shortcomings. The most striking shortcoming is the fact that the N2pc relies on lateral presentation of targets. For example, in Experiment 1 (Fig. 1, top panel), targets can appear both on the horizontal dimension and at the vertical dimension, but using the N2pc, one can only characterize the fingerprint of attentional selection on horizontally lateralized targets. As an alternative to the N2pc, one can use multivariate classification to characterize attentional selection, for example, classifying left versus right targets or classifying top versus bottom targets. These first-level classification analyses were performed when executing the initial script in the beginning of this chapter. The script below computes the associated group-level results of these classification analyses, both for the left versus right targets and for the top versus bottom targets. Finally, it plots these results in two separate graphs.

```
%% get group-level classification performance
cfg = [];
cfg.mpcompcor_method = 'cluster_based';
cfg.startdir = 'C:\EXP1\RESULTS';
cfg.timelim = [-100 400];
cfg.reduce_dims = 'diag';
mvpastats = adam_compute_group_MVPA(cfg); % press OK when the selection dialog pops up

%% plot classification performance over time for the LEFTRIGHT and the TOPBOTTOM dimension
cfg = [];
cfg.acclim = [.45 .7];
cfg.acctick = .05;
cfg.splinefreq = 32;
cfg.line_colors = {[0,0,0] [0,0,0]};
adam_plot_MVPA(cfg,mvpastats);
```

The first part of the script performs the group-level classification analysis on both contrasts: the horizontal dimension (left versus right target contrast) and the vertical dimension (top versus bottom target contrast). The analysis time window is restricted to $(-100, 400)$, specified in cfg.timelim. Classification performance is extracted for the diagonal, so without analyzing temporal generalization (specified in cfg.reduce_dims). This means that the data is trained and tested on the same samples. Details regarding the temporal generalization method are beyond the scope of this chapter, but more information can be found in other sources [5, 14]. The actual group-level analysis is performed by the function adam_compute_group_MVPA. When calling that function, one needs to specify the directory from which the data will be read, which in this case is the same directory as the cfg.startdir, so one can simply press OK after which group analyses from both analyses from contrasts are executed, performing t-testing against chance- level performance and applying cluster-based permutation to correct for multiple comparisons. The results are output in a variable called mvpastats, which has two elements: mvpastats(1) for the left-right classification analysis and mvpastats(2) for the top-bottom classification analysis.

Note that although testing against chance is common in the decoding literature, one caveat when using *t*-statistics on classification performance is that this does not allow population-level inference, in fact producing fixed effects rather than random effects results; see [15]. The implication is that one cannot formally draw population-level inferences based on such analyses, restricting conclusions to the sample that was tested. For studies that require population-level inference, it would be preferred to either use a completely separate training set (performing the training on different subjects or obtaining training data from a different task) or to replace the *t*-test with a statistic that explicitly evaluates information prevalence across the sampled subjects again; see [15].

In the second half of the script, the results in the mvpastats variable are plotted side by side, both in black outline. The cfg specifications have been explained before, except the splinefreq field. The splinefreq field causes the timeseries to be smoothed for visualization purposes. Smoothing is achieved by fitting a spline on the performance timeseries after downsampling it to 32 Hz (specified in cfg.splinefreq). The degree of smoothing is controlled by the resampling rate, with lower rates resulting in a smoother graph. The resampled series is centered on peak performance, so that the height of the peak is not affected by the smoothing procedure. This operation is applied for visualization purposes only; all statistical tests are performed on the unsmoothed timeseries.

The actual plotting operation is performed by the adam_plot_MVPA function. The result of the plotting operation is shown in the right two panels of Fig. 3. Note that these graphs no longer show μV on the y-axis, but instead show Area Under the Curve (AUC), a metric that indicates how well two or more classes can be discriminated by the classifier [16]. AUC typically runs between 0.5 (chance performance) and 1.0 (maximum classification accuracy). Interestingly, one can see from Fig. 3 how the two right panels show AUC time courses that are visually similar to the time course of the N2pc in the left panel. For targets on the horizontal meridian, this can be considered somewhat unsurprising, but a similar time course could not have been extracted using the standard N2pc approach for the vertical meridian. Thus, here, we see the first clear advantage of the multivariate classification approach.

As before, temporal onsets are automatically computed, reflecting the point in time where the rising signal reaches 50% of its peak amplitude [13]. As explained before, the exact values of the temporal onsets can be found by inspecting the latencies field of the mvpastats variable. Typing mvpastats.latencies returns two results, one for the left-right dimension (196 ms) and the other for the top-bottom dimension (204 ms). Note that the onset latency for the left-right is identical to the onset latency of the N2pc, and the latency for the top-bottom dimension is highly similar. One can test whether two onset latencies are different using jackknifing. Jackknifing is the practice of repeatedly computing an average while leaving out one subject, until each subject has been left out once. This way, you get the same number of observations as you have subjects in the dataset, but each observation is a group average with one of the subjects left out. This is useful when the single subject results are too noisy to determine a peak for every single subject (and thus to compute the 50% amplitude latency onset). The resulting jackknifed values can be used in a regular *t*-test or ANOVA, as long the resulting *t*- or *F*-values are corrected for jackknifing [17, 18]. This correction is applied automatically in the *t*-test function jackT from the latency package [13], which is included in the ADAM toolbox. Thus, to test whether the N2pc has a different onset latency from the left-right classification timeseries, one can simply type:

```
jackT(n2pcstats.latencies.jackknife, mvpastats(1).latencies.jackknife)
```

in the MATLAB command window. As a result, the function jackT will run a corrected t-test based on the jackknife latency onsets in the latencies fields of the n2pcstats and left vs. right classification in the mvpastats variables. Unsurprisingly, the result shows that the onset latencies between N2pc and classification are not significantly different; $t(11) = 0$, $p = 1$, providing converging evidence that the N2pc and the classification results tap into the same underlying signals. Similarly, one can also test whether the onsets between classifying left vs. right and classifying top vs. bottom are different by typing:

```
jackT(mvpastats(1).latencies.jackknife, mvpastats(2).latencies.jackknife)
```

Here too, the onset latencies are not significantly different between left vs. right and top vs. bottom; $t(11) = -1.62$, $p = 0.13$.

Multivariate classification seems the superior approach compared to the N2pc, as it allows one to characterize attentional selection both on the horizontal and on the vertical meridian. Moreover, it precludes one from having to perform a priori electrode selection. However, one might object that the downside of the approach is that it is hard to ascertain the source of the performance metric in the brain. Although the classifier produces training weights for the electrodes for every time sample, these classifier weights cannot be directly interpreted as neural sources [19]. Luckily, there are alternatives. The easiest way of characterizing the underlying cortical activity is to transform the classifier weights to forward weights by multiplying them with the data covariance matrix. Because the weights obtained from linear discriminant analysis contain the difference between the two compared sets normalized by the covariance matrix, this operation creates activation patterns that return the mass-univariate difference between the compared conditions, but which unlike classifier weights, are interpretable as neural sources [19]. Forward-transformed weights are equivalent to the univariate difference between conditions, except that they are derived from the classification analysis itself, providing a sanity check that the classification analysis results in a meaningful pattern of results. The ADAM toolbox automatically computes forward-transformed weights during the first level and stores these in the output variable during group-level analysis. The script to plot the forward-transformed weights for the two classification analyses (right versus left and top versus bottom) is given below.

```
%% plot topographic maps
cfg = [];
cfg.plotweights_or_pattern = 'covpattern';
cfg.timelim = [240 250];
cfg.weightlim = [-1.8 1.8];
cfg.mpcompcor_method = 'cluster_based';
adam_plot_BDM_weights(cfg,mvpastats);
```

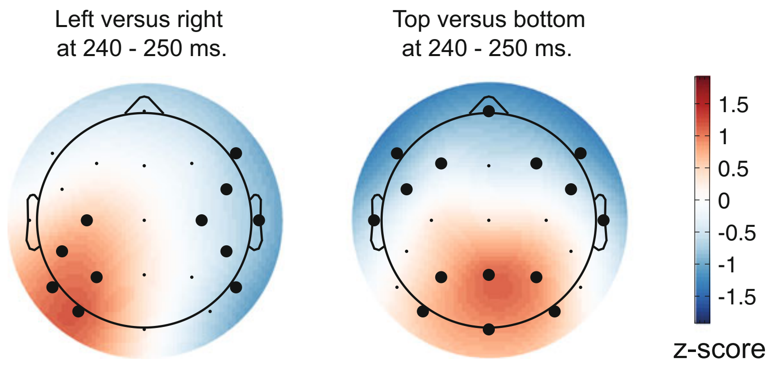

Fig. 4 Activation patterns associated with peak decoding accuracy (240–250 ms) in Experiment 1. Derived from the product of the weight vectors and the covariance matrix, normalized across space (see main text). *Left*: The pattern associated with left versus right decoding. Note the clearly lateralized distribution. This lateralized pattern shows the distribution of neural activity underlying successful discrimination between targets appearing on the left and the right of fixation and is equivalent to the mass-univariate difference between left and right targets. *Right*: The pattern associated with top versus bottom decoding, now showing a posterior-anterior distribution. Thick electrode dots belong to clusters having $p < 0.05$ under cluster-based permutation testing

The adam_plot_BDM_weights function computes two topographic plots from the mvpastats data, which can be found in Fig. 4. The cfg variable first specifies that the function should plot the pattern based on the covariance matrix (indicated in cfg.plotweights_or_pattern). Further, it averages the plot in the temporal window between 240 and 250 ms (specified in cfg.timelim), approximately corresponding to the peak of the performance timeseries. Each of the electrodes is tested against zero using a *t*-test, after which a cluster-based permutation test is executed based on the adjacency of neighboring electrodes to correct for multiple comparisons. Electrodes that survive the cluster-based permutation test are indicated as thick dots on topographic map. Note that the adam_plot_BDM_weights function spatially normalizes the pattern for every subject prior to computing the group average, so that the amplitude at every electrode is expressed as a *Z*-score across electrodes.

To further highlight the advantage of multivariate classification to characterize attentional selection, we now move our attention to the second experiment. In this experiment, targets were not presented on the horizontal or on the vertical meridian, but rather in a circular array (see Fig. 1, bottom). Here, we first ask the question whether it is possible to characterize attentional selection without even crossing the meridian, so, for example, within a quadrant.

For example, we may ask whether one can dissociate attentional selection of targets within each of the quadrants of the visual field, such as between targets on position 1 and targets on position 2, between position 3 and 4, etc. (see Fig. 1, right bottom). The script to execute the first level decoding analyses to extract these analyses is given below.

```
%% general information about experiment 2
filenames = { 'DecExp3_1_R'  'DecExp3_2_G'  'DecExp3_3_B'  'DecExp3_4_R'  'DecExp3_5_G' ...
              'DecExp3_6_B'  'DecExp3_7_R'  'DecExp3_8_G'  'DecExp3_9_B'  'DecExp3_10_R' ...
              'DecExp3_11_G' 'DecExp3_12_B' 'DecExp3_13_R' 'DecExp3_14_G' 'DecExp3_15_B' };
for c=1:8
    pos{c} = [ 10+c 20+c ];
end

%% settings for first level quadrant analysis of experiment 2
cfg = [];
cfg.datadir = 'C:\EXP2\EEGDATA';
cfg.filenames = filenames;
cfg.erp_baseline = [-.1,0];
cfg.resample = 250;
cfg.nfolds = 10;
cfg.model = 'BDM';
cfg.channels = 'ALL';

% classify attentional selection in the upper right quadrant
clear class_spec;
class_spec{1} = cond_string(pos{1});
class_spec{2} = cond_string(pos{2});
cfg.class_spec = class_spec;
cfg.outputdir = 'C:\EXP2\RESULTS\QUADRANT\1_2';
adam_MVPA_firstlevel(cfg);

% classify attentional selection in the bottom right quadrant
clear class_spec;
class_spec{1} = cond_string(pos{3});
class_spec{2} = cond_string(pos{4});
cfg.class_spec = class_spec;
cfg.outputdir = 'C:\EXP2\RESULTS\QUADRANT\3_4';
adam_MVPA_firstlevel(cfg);

% classify attentional selection in the bottom left quadrant
clear class_spec;
class_spec{1} = cond_string(pos{5});
class_spec{2} = cond_string(pos{6});
cfg.class_spec = class_spec;
cfg.outputdir = 'C:\EXP2\RESULTS\QUADRANT\5_6';
adam_MVPA_firstlevel(cfg);

% classify attentional selection in the upper left quadrant
clear class_spec;
class_spec{1} = cond_string(pos{7});
class_spec{2} = cond_string(pos{8});
cfg.class_spec = class_spec;
cfg.outputdir ='C:\EXP2\RESULTS\QUADRANT\7_8';
adam_MVPA_firstlevel(cfg);
```

The above script performs analyses analogous to the analysis that was performed in Experiment 1, but now within the four quadrants of Experiment 2. The group-level analyses can be executed and plotted using the script below.

```
%% get group-level classification performance within each quadrant quadrant, experiment 2
cfg = [];
cfg.startdir = 'C:\EXP2\RESULTS';
cfg.mpcompcor_method = 'cluster_based';
cfg.timelim = [-100 400];
cfg.reduce_dims = 'diag';
cfg.channelpool = 'ALL';
cfg.plotmodel = 'BDM';
mvpastats_quadrant = adam_compute_group_MVPA(cfg); % select the folder QUADRANT

%% plot classification performance over time for each of the four quadrants
cfg = [];
cfg.acclim = [.46 .7];
cfg.acctick = .05;
cfg.splinefreq = 32;
cfg.line_colors = {[0,0,0],[0,0,0],[0,0,0],[0,0,0]};
cfg.plotorder = {'7_8' '1_2' '5_6' '3_4' };
cfg.nolatency = true;
adam_plot_MVPA(cfg,mvpastats_quadrant);
```

The first part of the script once again reads in the first level results and computes the group-level results. When the function adam_compute_group_MVPA is executed, a dialog appears. Select "QUADRANT" to read in the analyses from the four different quadrants. These will be stored in the variable mvpastats_quadrant, the contents of which can subsequently be plotted using adam_plot_MVPA. The cfg that is used to plot the results is much the same as before, with two minor additions. A field plot_order was added to control the order in which the analyses are plotted. The names in plot_order are taken directly from the folder names that are used to store the different first-level analyses. Further, a field nolatency is used to preclude the plotting of latency information. Although the graphs in Fig. 5 clearly show that it is possible to classify which item was attentionally selected within each quadrant, eyeballing the data already suggests that classifier performance is not reliable enough over time to estimate consistent onset latencies (e.g., see the top left panel in Fig. 5). For this reason, I chose not to plot latency information in this graph.

Indeed, to reliably determine onset latency, the analysis would require more data. The search display has eight target positions, so to increase the signal-to-noise ratio, one can perform an 8-way classification analysis, inputting each of the eight positions as classes into the classification analysis. This should provide the best possible estimate in Experiment 2 of the time course of attentional selection across the display and is achieved using the script below (keeping the same filename and condition definitions as in the quadrant analysis above).

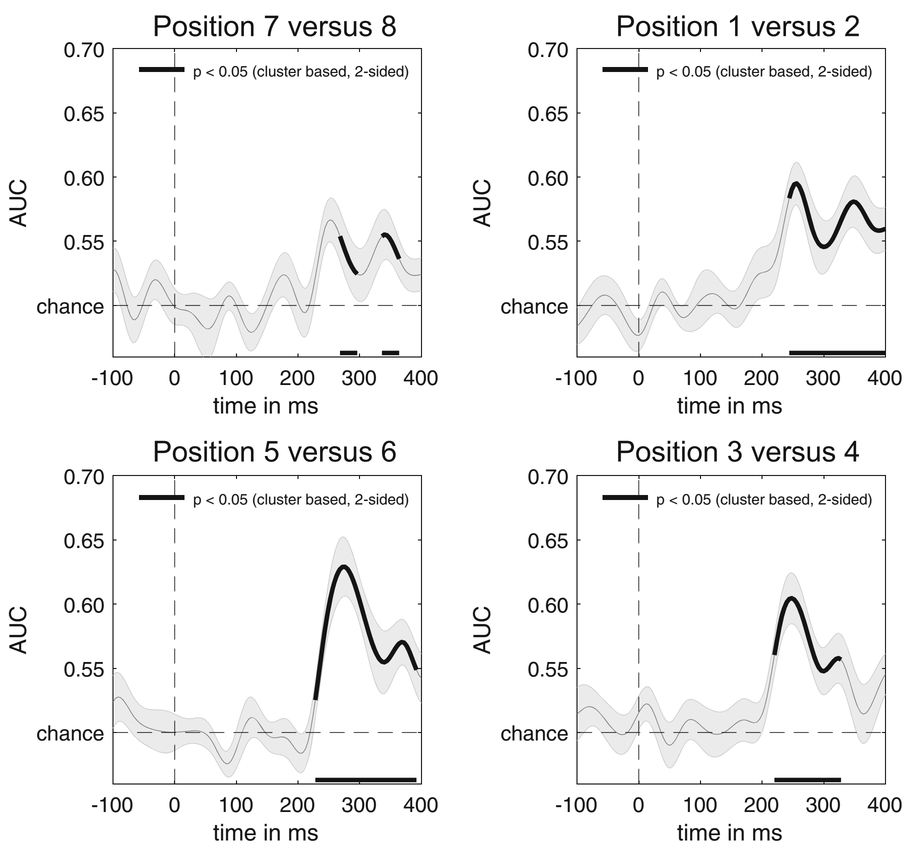

Fig. 5 Per quadrant decoding accuracy of target position in Experiment 2. Thick black lines reflect statistical tests that survive cluster-based permutation testing at $p < 0.05$. Shaded areas are ±s.e.m

```
%% settings for first level analysis of all positions in experiment 2
cfg = [];
cfg.datadir = 'C:\EXP2\EEGDATA';
cfg.filenames = filenames;
cfg.model = 'FEM,BDM';
cfg.resample = 250;
cfg.channels = 'ALL';
cfg.erp_baseline = [-.1,0];
cfg.sigma_basis_set = 0;

% classify attentional selection across all eight target positions
clear class_spec;
for c = 1:8
    class_spec{c} = cond_string(pos{c});
end
cfg.class_spec = class_spec;
cfg.outputdir = 'C:\EXP2\RESULTS\ALLPOS';
adam_MVPA_firstlevel(cfg);
```

Note that the above script not only runs a decoding analysis ('BDM' specified in cfg.model) but also a forward encoding analysis ('FEM' specified in cfg.model). We will return to this analysis in the next section. But before we do so, we first plot the result of the 8-way decoding analysis and determine the concomitant onset latency from that analysis. Below is the script to compute the group results and plot the 8-way classification analysis.

```
%% get group-level 8-way classification performance, experiment 2
cfg = [];
cfg.startdir = 'C:\EXP2\RESULTS';
cfg.mpcompcor_method = 'cluster_based';
cfg.timelim = [-100 400];
cfg.reduce_dims = 'diag';
cfg.channelpool = 'ALL';
cfg.plotmodel = 'BDM';
mvpastats_8way = adam_compute_group_MVPA(cfg); % select the folder ALLPOS

%% plot 8-way decoding
cfg = [];
cfg.acclim = [.45 .75];
cfg.acctick = .05;
cfg.splinefreq = 32;
adam_plot_MVPA(cfg,mvpastats_8way);
```

When the adam_compute_group_MVPA function executes, a folder selection window pops up in which one should manually select the ALLPOS folder. This folder contains the first-level results for all 8-way classification of the eight target positions. As before, this will compute the group results and assign this outcome to a variable (mvpastats_8way), subsequently plotting this outcome using the adam_plot_MVPA function. The resulting plot can be found in Fig. 6 below, which now also highlights the 50% peak amplitude onset latency using a vertical dotted line. The numerical value associated with the 50% peak amplitude onset latency in this plot can be found by typing mvpastats_8way.latencies.GA, which returns 220 ms. Interestingly, this onset latency seems slightly later

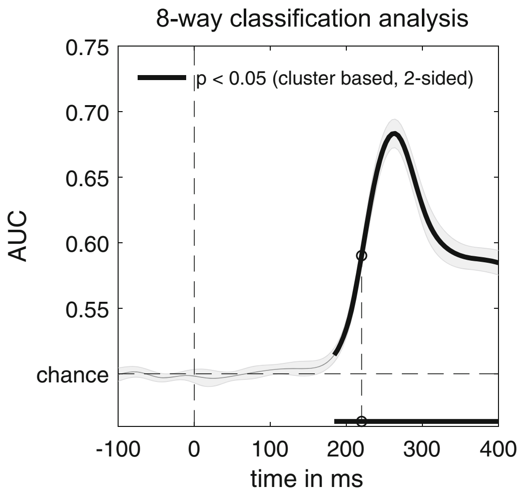

Fig. 6 8-way classification accuracy of target position in Experiment 2. Note again the similarity to the temporal evolution of the N2pc (Fig. 3, left) and decoding performance in Experiment 1 (Fig. 3, middle and right), although plausibly having a slightly later onset because of the increase in the number of items on the screen. Thick black lines reflect statistical tests that survive cluster-based permutation testing at $p < 0.05$. The shaded area is $\pm$s.e.m

than the onset latencies that were identified in Experiment 1, possibly because of the larger number of items in the display. Determining the true cause of this apparent latency difference is beyond the scope of the current chapter and would require further experimentation. I suffice to point out here that one can investigate such onset latency differences using the methods that are explained in this chapter.

3 Characterizing the N2pc Using Forward Encoding

So far, this chapter has covered classification approaches to characterize attentional selection. In this section, I discuss a complementary multivariate approach, which is to use a forward encoding model to establish a continuous relationship between an experimental variable of interest and cortical activity [4, 20, 21]. This approach has been further extended using inverted encoding models that estimate model responses from the data in so-called channel tuning functions (CTFs); e.g., see [22–25]. Below, I first describe the general approach that is taken in these models, and I discuss some caveats of the method [26–28]. Next, I provide details and script that applies a forward encoding model (FEM) to the data obtained from Experiment 2 and show how it can be used to reconstruct cortical activity for conditions that did not occur during the experiment.

The principal goal of forward encoding models is to characterize a direct link between a continuous stimulus parameter space and the cortical responses that are measured (here through EEG). The advantage of this approach is that one can predict (reconstruct) cortical activations for novel stimulus values that were never presented during the experiment or stimulus parameter estimates for novel brain data for which no condition labels were acquired [20]. In addition, a number of studies have suggested that one can use inverted encoding models to estimate the model response (referred to as channel tuning function, or CTF), to assay of how broad-scale cortical activity is tuned to a continuous experimental variable, somewhat akin to neural tuning functions at the level of single neurons [29].

Despite their initial promise, it has recently been shown that the width of a CTF not only reflects the degree of tuning to the parameter space but also the signal-to-noise ratio of the data on which the model is fitted [27]. Moreover, it has been shown that such a fitting procedure recovers arbitrary starting parameters of the model, rather than a recovering a CTF that reflects the actual relationship between the experimental variable and cortical activity [26]. With these caveats in mind, one might still use the simplest form of these models (a delta function) to construct a CTF to provide insight into the degree to which neighboring experimental parameter values produce overlapping cortical activations.

Here, we generate a FEM of the data in the second experiment (Fig. 2, bottom). To do so, we employ a procedure previously described by Brouwer and Heeger [20] using the same tenfold cross validation scheme as in the previously described classification analyses. In this procedure, the training set is used to estimate the response in each of eight hypothetical position "channels" (corresponding to the eight target positions on the screen). The nomenclature "channels" here should not be confused with MEG or EEG sensors; EEG sensors are referred to as electrodes in the current chapter. To provide an initial estimate of the channel responses, a preliminary "basis set" is used to estimate the weights that specify the relationship between the observed multivariate signal and the channel responses. Typically, authors have used a basis set in the form of a Gaussian or a sinusoid raised to a power, but as explained above, it has recently been shown that CTF estimation using this method can recover any arbitrary basis set, rather than assaying the true relationship between the continuous experimental variable under investigation and the measured multivariate activity [26].

For this reason, I recommend here to only use the simplest form of the basis set, containing a 1 for the corresponding target position and a 0 for all other positions, so that the shape of any resulting CTF cannot reflect the initial basis set, but must be caused by the data itself. A binary on-off basis set like this is sometimes also referred to as a delta function. Here, we use eight basis sets (one for each target position), each shifted by one position compared to its neighbor, to construct a regression matrix C1. C1 has the form $k \times n1$, in which k is the number of position channels (1 to 8) and n1 is the number of trials in the training set. Next, we estimate the response amplitude to each of the eight hypothetical position channels by performing an ordinary least squares regression of the C1 matrix onto the B1 matrix from the EEG training set. B1 contained EEG data of the form $m \times n1$, in which m is number of electrodes and n1 is the number of trials in the training set. This regression yields a weight matrix W in which each electrode obtains a regression coefficient (a "weight") for each hypothetical channel. The weight matrix W has the form $m \times k$, in which m is the number of electrodes and k is the number of position channels.

Next, the model is inverted by performing ordinary least squares regression of these weights onto the B2 matrix from the EEG testing set to produce the estimated channel responses for each trial. B2 has the form $m \times n2$, in which m is number of electrodes and n2 is the number of trials in the testing set. The resulting estimated channel responses are contained in matrix C2, having the form $k \times n2$, in which k are the observed channel responses and n2 are the trials in the testing set. This procedure is repeated for all folds in the train-test procedure, until all data has been tested once. Next, the channel responses are averaged across trials in the testing set, separately for each of the eight trial types that correspond to each of the eight target positions on the screen.

The channel responses from this testing phase in combination with the associated weights contain the validated and invertible one-to-one relationship between a particular attended location in the search display and the multivariate EEG response. The script to perform the above procedure was executed when specifying 'FEM' during the first-level analyses of Experiment 2. The (averaged) C2 channel responses constitute a CTF per condition. These can be shifted to a common center, so that the channel responses for each of the eight target positions are aligned and averaged to obtain a canonical CTF. Mathematical (less verbal) descriptions as well as more graphical depictions of this train-test estimation procedure have been provided elsewhere, e.g., [20–28]. The script below extracts and plots the CTFs from the first-level analyses.

```
%% compute FEM in experiment 2
cfg = [];
cfg.startdir = 'C:\EXP2\RESULTS';
cfg.mpcompcor_method = 'cluster_based';
cfg.plotmodel = 'FEM';
cfg.channelpool = 'ALL';
cfg.timelim = [-100,1000];
cfg.reduce_dims = 'diag';
femstats = adam_compute_group_MVPA(cfg); % select the folder ALLPOS

%% plot CTF at 260-270 ms
cfg = [];
cfg.plotfield = 'CTFpercond';
cfg.shiftindiv = true;
cfg.weightlim = [-.2 .6];
cfg.CTFtime = [260 270];
cfg.BLtime = [-100 0];
CTF = adam_plot_CTF(cfg,femstats);
```

First, it extracts the group-level channel responses and corresponding weights using adam_compute_group_MVPA (cfg. plotmodel as 'FEM') and stores these results in a stats variable called femstats. Next, it uses the function adam_plot_CTF to plot the CTF for each condition (specified in cfg.plotfield as 'CTFpercond') in the period between 260 and 270 ms for the CTF (specified in cfg.CTFtime, corresponding the peak classification accuracy in Fig. 6) as well as the CTF in the baseline between −100 and 0 ms (specified in cfg.BLtime). The resulting figure is shown in the top left panel of Fig. 7. These CTFs are taken from the condition-specific averages of C2, but note that these responses are shifted to a common center, so that the channel responses for each of the eight target positions are aligned. This is specified by indicating ctf. shiftindiv = true (see labels under the x-axis of the figure to see how they were shifted). One can also plot the average of these shifted condition-specific CTFs to show the canonical CTF. This can be

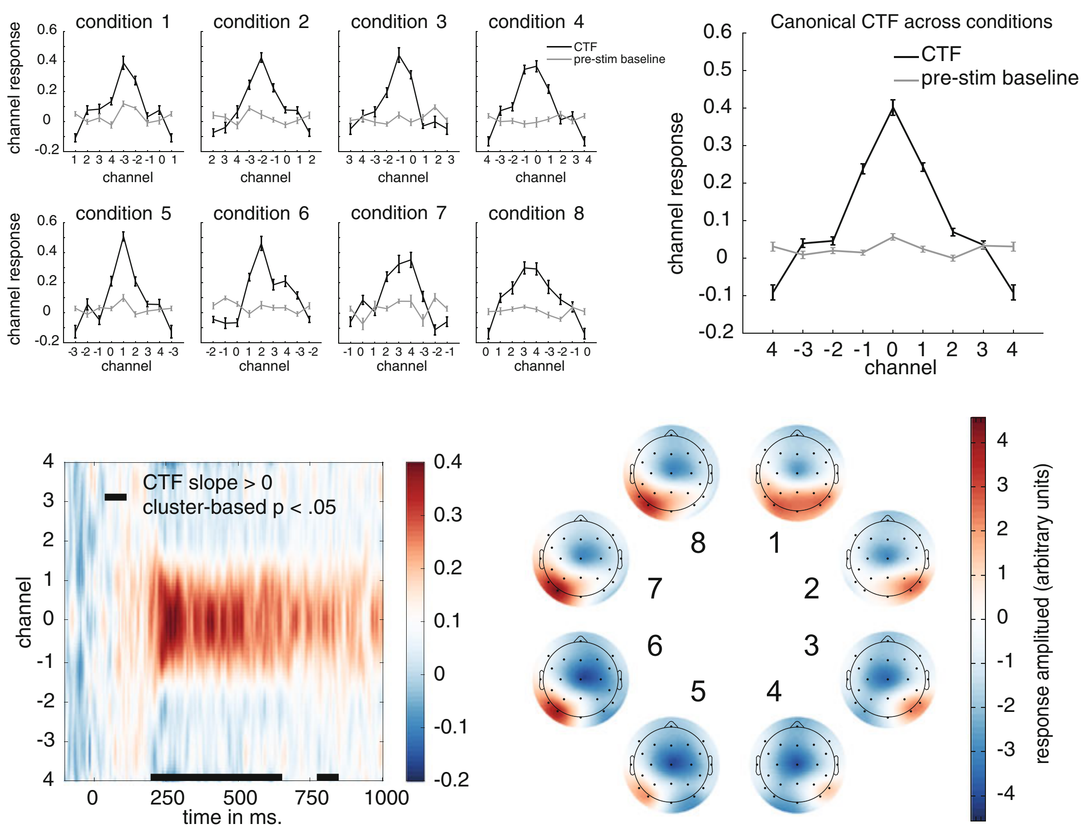

Fig. 7 Channel tuning functions. *Top left*: CTFs for individual conditions show that the CTF is not driven by particular target positions. *Top right*: CTF for the 260–270 ms window and CTF during baseline (−100 to 0 ms) obtained by shifting the individual condition CTFs to align to the same channel. *Bottom left*: CTF development over time in which color reflects channel responses. The black line near the time axis shows the time windows where the slope of the CTF is significantly different from 0 ($p < 0.05$, cluster-based permutation test). *Bottom right*: Topographic weight plots for each condition in the 260–270 ms time window. Weights from forward models are directly interpretable in terms of neural sources. These plots therefore show how neural activity changes as a function of variability in attended target position

done by running the same script as above, the only difference being that cfg.plotfield should be changed to 'CTF' prior to running the script, like this:

```
cfg.plotfield = 'CTF';
```

Figure 7 top right shows the averaged empirical CTF across conditions. Importantly, we used a basis set that did not make any assumptions about the shape of the CTF beforehand (the delta function), so we can be sure that the CTF reflects the relationship between an experimental parameter of interest (the hypothetically attended location channel, on the x-axis) and the strength of the multivariate response (on the y-axis). What this

CTF shows is that there is some degree of overlap between multivariate responses for neighboring attended locations, with the caveat that the exact strength of this overlap as quantified by the CTF is also affected by the signal-to-noise ratio [27]. Note further that in the top of Fig. 7, we plotted the CTFs during peak classification performance, between 260 and 270 ms. However, the estimation procedure was done for every time sample, yielding a CTF over time. The CTF over time can also be plotted, using the script below.

```
%% plot CTF over time
cfg = [];
cfg.plotfield = 'CTF';
cfg.reduce_dims = 'diag';
cfg.colorlim = [-.2, .4];
adam_plot_CTF(cfg,femstats);
```

The script is the same as before, the only difference being that the we no longer specify CTFtime, so that the function plots the CTF over the entire time interval rather than averaging over a time window, now using color to denote the strength of the channel response for every time point, and plotting the channels on the y-axis. The resulting plot can be found in Fig. 7, left bottom. Aside from plotting CTFs, one might also be interested in knowing the topographic distribution of these responses. Fortunately, the weights resulting from a forward encoding model can be interpreted directly as a neural source [19]. The script to plot the weights for each of the eight target positions is given below.

```
%% plot FEM weights, experiment 2
cfg = [];
cfg.timelim = [260 270];
cfg.mpcompcor_method = 'none';
cfg.normalized = false;
adam_plot_FEM_weights(cfg,femstats);
```

This produces the topographic weight maps for each of the eight target positions. This is shown in Fig. 7, bottom right. Note that no statistics are applied (cfg.mpcompcor_method as 'none'), and the plots are not spatially normalized (cfg. normalized = false).

Finally, we establish how a forward encoding model can be used to reconstruct cortical activity for experimental stimulus values that were used to generate the model. The top left of Fig. 8 shows the target display from Experiment 2, containing four target positions that were not present in the experiment (top, bottom, left, and right position). The aim of the following section is to reconstruct cortical activity associated with these positions using the specified forward model. The first step in this reconstruction is to construct CTFs (channel responses) that would have occurred when these positions would have

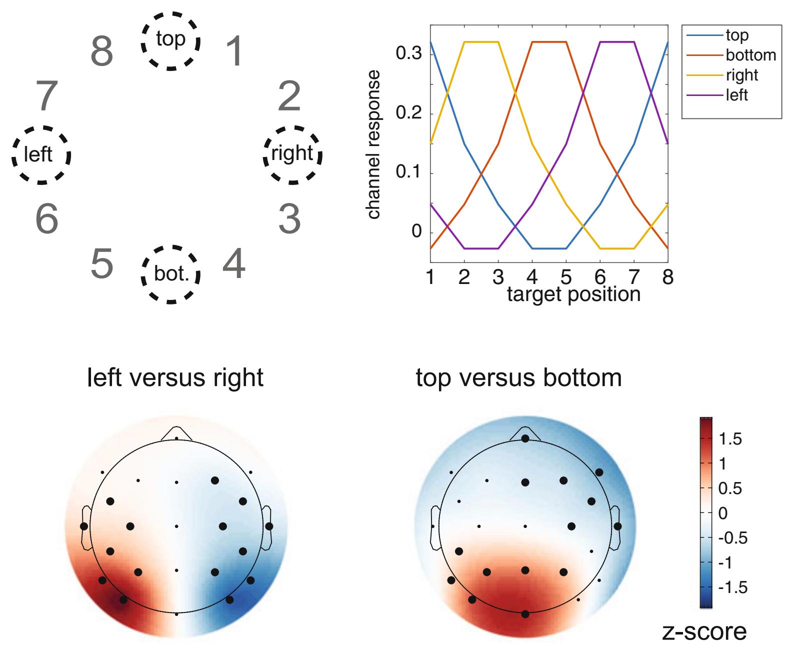

Fig. 8 Reconstructing the neural signature of attentional capture for target positions that were never presented during Experiment 2. *Top left*: The target positions that are reconstructed: top, bottom, left, and right. *Top right*: The constructed channel responses that are associated with these positions using the CTF from Fig. 7 (see main text for details). Top is in between target position 8 and 1, so the channel response to top is constructed by averaging the hypothetical channel response to position 8 and position 1. Similarly, right is created from averaging channel responses to 2 and 3, etc. Any position on the circle can be constructed using a weighted average of channel responses. Left, right, bottom, and top weights were reconstructed using the product of the constructed channel responses and the weight matrix at 260–270 ms. *Bottom left*: The left versus right pattern was generated by subtracting the left from the right pattern. *Bottom right*: The top versus bottom pattern was created by subtracting the bottom pattern from the top pattern. Note the similarity with the left-right and top-bottom patterns from Experiment 1. Patterns are normalized across electrodes. Thick electrode dots survived cluster-based permutation testing under $p < 0.05$

occurred in the experiment. For the top position, this would be the interpolated CTF between location 8 and 1. For the right position, this would be the interpolated CTF between location 2 and 3 and so forth. These CTFs are generated by taking the canonical CTF and using it to generate these interpolated CTFs. The CTF values during peak classification response were returned when plotting that CTF above (CTF = adam_plot_CTF(cfg,fem-stats) in the script). These values are used below to create the channel-specific CTFs.

```
%% create new CTFs from the canonical CTF
basis_set = mean(CTF.indivCTFmean); % a basis set created from the canonical CTF

% mirror
basis_set = (basis_set + basis_set(end:-1:1))/2;
basis_set = basis_set(2:9); % remove duplicate end point

% generate channel responses for each condition (shifted)
old_chan_responses = nan(numel(basis_set),numel(basis_set));
for c=1:numel(basis_set)
    old_chan_responses(:,c) = circshift(shiftdim(basis_set),-floor(numel(basis_set)/2)+c);
end
% interpolate new channel response sets for positions between 1 and 8, between 4
% and 5 (the vertical ones) and between 2 and 3 and 6 and 7 (the horizontal ones)
new_chan_responses = nan(numel(basis_set),4);
new_chan_responses(:,1) = mean([ old_chan_responses(:,8) old_chan_responses(:,1) ],2);
new_chan_responses(:,2) = mean([ old_chan_responses(:,4) old_chan_responses(:,5) ],2);
new_chan_responses(:,3) = mean([ old_chan_responses(:,2) old_chan_responses(:,3) ],2);
new_chan_responses(:,4) = mean([ old_chan_responses(:,6) old_chan_responses(:,7) ],2);

% plot the result
figure; plot(new_chan_responses);
```

These channel responses for top, bottom, left, and right are plotted in Fig. 8, top right. Next, these interpolated channel responses are used to generate new weight matrices for these positions. This is done in the first section of the script below.

```
%% reconstruct patterns, respectively for top, bottom, right and left
for cSubj = 1:size(femstats.weights.indivWeights,1)                % subject loop
    for cT = 1:size(femstats.weights.indivWeights,2)               % time loop
        W = squeeze(femstats.weights.indivWeights(cSubj,cT,:,:));  % extract channel weights
        indivWeights(cSubj,cT,:,:) = W*new_chan_responses;         % new weights
    end
end

%% subtract channel weights on the horizontal and vertical meridian
newIndivWeights(:,:,:,1) = indivWeights(:,:,:,4) - indivWeights(:,:,:,3); % bottom from top
newIndivWeights(:,:,:,2) = indivWeights(:,:,:,1) - indivWeights(:,:,:,2); % left from right

%% insert new subtractions into femstats for plotting
constructed_femstats = femstats;                             % copy what we had
constructed_femstats.weights.indivWeights = newIndivWeights; % inject new weights

% plot reconstructed patterns
cfg = [];
cfg.timelim = [260 270];
cfg.weightlim = [-1.8 1.8];
cfg.mpcompcor_method = 'cluster_based';
adam_plot_FEM_weights(cfg,constructed_femstats);
```

The second section of this script subtracts the bottom from the top and the left from the right position, to get the distribution associated with top versus bottom and left versus right. Finally, these new weight matrices are injected into a femstats variable for plotting using the adam_plot_FEM_weights function that we used before. The resulting topographic maps are plotted in Fig. 8, bottom. If these topographic plots look familiar, that is no coincidence. They seem to nicely correspond to the topographic maps that were obtained in the first experiment (Fig. 4). However, a crucial difference between Figs. 4 and 8, bottom, is that the topographies in Fig. 4 were derived from actual data in Experiment 1, whereas the topographies in Fig. 8 were constructed from the forward encoding model and do not correspond to actual data that was collected during Experiment 2. The correlation between the topographic maps from Experiment 1 and Experiment 2 is extremely high ($r = 0.85$, $p < 10^{-6}$ for the horizontal meridian and $r = 0.86$, $p < 10^{-6}$ for the veridical meridian), thus providing converging evidence that the forward encoding model is able to successfully construct cortical activation maps for data that was not actually present in the data that was used to generate the forward encoding model.

4 Conclusion

This chapter compared univariate to multivariate methods when analyzing EEG data obtained during tasks in which subjects need to use feature-based attention to select items in a display. This shows that multivariate classification is superior to traditional univariate analysis when characterizing the spatiotemporal profile of attentional selection. Experiment 1 shows how multivariate classification analyses enable one to not only assess attentional selection on the horizontal meridian but also on the vertical meridian. Experiment 2 shows that one can use classification analyses to assess the time course of attentional selection within quadrants of the visual field and that one can even look at the time course of attentional selection across a large number of attended positions. Further, Experiment 2 shows how one can use a forward modeling approach to construct spatiotemporal responses for locations that were never attended during the experiment. The chapter has also demonstrated how one can assess onset latencies, as well as how one can plot spatiotemporal maps of both decoding and forward encoding analyses. Together, this should provide a useful introduction for those in the field of feature-based attentional selection that want to move from traditional univariate analysis to multivariate analysis.

Acknowledgments

I would like to thank Anna Grubert, Martin Eimer, and Chris Olivers for allowing me to freely use and share the data from these experiments, as well as the analysis plans that we used on these data. This chapter would not have existed without them.

References

1. Eimer M (1996) The N2pc component as an indicator of attentional selectivity. Electroencephalogr Clin Neurophysiol 99(3):225–234. https://doi.org/10.1016/0013-4694(96)95711-9
2. Luck SJ, Hillyard SA (1994) Electrophysiological correlates of feature analysis during visual search. Psychophysiology 31(3):291–308
3. Woodman GF (2010) A brief introduction to the use of event-related potentials (ERPs) in studies of perception and attention. Atten Percept Psychophys 72(8):2031–2046. https://doi.org/10.3758/APP.72.8.2031
4. Fahrenfort JJ, Grubert A, Olivers CNL, Eimer M (2017) Multivariate EEG analyses support high-resolution tracking of feature-based attentional selection. Sci Rep 7(1):1886. https://doi.org/10.1038/s41598-017-01911-0
5. Fahrenfort JJ, van Driel J, van Gaal S, Olivers CNL (2018) From ERPs to MVPA using the Amsterdam decoding and modeling toolbox (ADAM). Front Neurosci 12. https://doi.org/10.3389/fnins.2018.00368
6. Grootswagers T, Wardle SG, Carlson TA (2017) Decoding dynamic brain patterns from evoked responses: a tutorial on multivariate pattern analysis applied to time series neuroimaging data. J Cogn Neurosci 29(4):677–697. https://doi.org/10.1162/jocn_a_01068
7. van Driel J, Olivers CNL, Fahrenfort JJ (2019) High-pass filtering artifacts in multivariate classification of neural time series data. bioRxiv. https://doi.org/10.1101/530220
8. VanRullen R (2011) Four common conceptual fallacies in mapping the time course of recognition. Front Psychol 2:365. https://doi.org/10.3389/fpsyg.2011.00365
9. Maris E, Oostenveld R (2007) Nonparametric statistical testing of EEG- and MEG-data. J Neurosci Methods 164(1):177–190. https://doi.org/10.1016/J.Jneumeth.2007.03.024
10. Sassenhagen J, Draschkow D (2019) Cluster-based permutation tests of MEG/EEG data do not establish significance of effect latency or location. Psychophysiology 35(2):e13335. https://doi.org/10.1111/psyp.13335
11. Kiesel A, Miller J, Jolicoeur P, Brisson B (2008) Measurement of ERP latency differences: a comparison of single-participant and jackknife-based scoring methods. Psychophysiology 45(2):250–274. https://doi.org/10.1111/j.1469-8986.2007.00618.x
12. Luck SJ (2014) An introduction to the event-related potential technique. MIT Press, Cambridge, MA. https://doi.org/10.1086/506120
13. Liesefeld HR (2018) Estimating the timing of cognitive operations with MEG/EEG latency measures: a primer, a brief tutorial, and an implementation of various methods. Front Neurosci 12:765. https://doi.org/10.3389/fnins.2018.00765
14. King JR, Dehaene S (2014) Characterizing the dynamics of mental representations: the temporal generalization method. Trends Cogn Sci 18(4):203–210. https://doi.org/10.1016/j.tics.2014.01.002
15. Allefeld C, Görgen K, Haynes J-D (2016) Valid population inference for information-based imaging: from the second-level t-test to prevalence inference. Neuroimage 141:378–392. https://doi.org/10.1016/j.neuroimage.2016.07.040
16. Hand DJ, Till RJ (2001) A simple generalisation of the area under the ROC curve for multiple class classification problems. Mach Learn 45(2):171–186. https://doi.org/10.1023/A:1010920819831
17. Miller J, Patterson T, Ulrich R (1998) Jackknife-based method for measuring LRP onset latency differences. Psychophysiology 35(1):99–115
18. Ulrich R, Miller J (2001) Using the jackknife-based scoring method for measuring LRP onset effects in factorial designs. Psychophysiology 38(5):816–827. https://doi.org/10.1111/1469-8986.3850816
19. Haufe S, Meinecke F, Goergen K, Daehne S, Haynes J-D, Blankertz B, Biessgmann F (2014) On the interpretation of weight vectors of linear models in multivariate neuroimaging. Neuroimage 87:96–110. https://doi.org/10.1016/j.neuroimage.2013.10.067
20. Brouwer GJ, Heeger DJ (2009) Decoding and reconstructing color from responses in human visual cortex. J Neurosci 29(44):13992–14003. https://doi.org/10.1523/JNEUROSCI.3577-09.2009
21. Garcia JO, Srinivasan R, Serences JT (2013) Near-real-time feature-selective modulations in human cortex. Curr Biol 23(6):515–522. https://doi.org/10.1016/j.cub.2013.02.013
22. Ester EF, Sprague TC, Serences JT (2015) Parietal and frontal cortex encode stimulus-specific mnemonic representations during

visual working memory. Neuron 87 (4):893–905. https://doi.org/10.1016/j.neuron.2015.07.013

23. Foster JJ, Sutterer DW, Serences JT, Vogel EK, Awh E (2017) Alpha-band oscillations enable spatially and temporally resolved tracking of covert spatial attention. Psychol Sci 28 (7):929–941. https://doi.org/10.1177/0956797617699167
24. Foster JJ, Sutterer DW, Serences JT, Vogel EK, Awh E (2016) The topography of alpha-band activity tracks the content of spatial working memory. J Neurophysiol 115(1):168–177. https://doi.org/10.1152/jn.00860.2015
25. Samaha J, Sprague TC, Postle BR (2016) Decoding and reconstructing the focus of spatial attention from the topography of alpha-band oscillations. J Cogn Neurosci 28 (8):1090–1097. https://doi.org/10.1162/jocn_a_00955
26. Gardner JL, Liu T (2019) Inverted encoding models reconstruct an arbitrary model response, not the stimulus. eNeuro 6(2). pii: ENEURO.0363-18.2019. https://doi.org/10.1523/ENEURO.0363-18.2019
27. Liu T, Cable D, Gardner JL (2018) Inverted encoding models of human population response conflate noise and neural tuning width. J Neurosci 38(2):398–408. https://doi.org/10.1523/JNEUROSCI.2453-17.2017
28. Sprague TC, Adam KCS, Foster JJ, Rahmati M, Sutterer DW, Vo VA (2018) Inverted encoding models assay population-level stimulus representations, not single-unit neural tuning. eNeuro 5(3). pii: ENEURO.0098-18.2018. https://doi.org/10.1523/eneuro.0098-18.2018
29. Hubel DH, Wiesel TN (1962) Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J Physiol 160:106–154