Demonstration scripts for "Multiway Canonical Correlation Analysis of Brain Signals".

These scripts can be found at http://audition.ens.fr/adc/NoiseTools/EXAMPLES/MCCA\_EXAMPLE\_SCRIPTS/

How do you implement MCCA?

* mcca\_demo1.m

shows how to call routines from Alain de Cheveigné's NoiseTools toolbox (http://audition.ens.fr/adc/NoiseTools/) or from Lucas Parra's code for MCCA (https://www.parralab.org/corrca/, or simple matlab code.

What does the outcome of MCCA look like?

* mcca\_demo2.m

shows results for MCCA applied to a synthetic data set (1 target shared source mixed into 5 data matrices full of noise). Fig 1 shows the target and one channel of one data matrix of the mixture: the target is lost in the noise. Fig 2 shows the result of applying MCCA: the first SC (corresponding to the recovered shared target) has higher variance than the rest (left) and its time course (right) looks like the target.

MCCA produces both summary components (SC) and canonical correlates (CC). What is the difference? Summary components (SC) summarize the entire data set, and result from a transform applied to the concatenation of the data matrices. Canonical correlates (CCs) result from a transform applied to one matrix (e.g. subject).

* mcca\_demo2b.m

shows both (for data similar to mcca\_demo2.m). The CC transform matrices are slices of the SC transform matrix (Fig 1). Each SC is the sum of all CCs of same rank. Note the SNR in this example: MCCA can extract very weak sources in the ideal case where target and noise are linearly separable. See mcca\_demo6.m for the (more realistic) case where target and noise are not linearly separable. Also the case for real (non-synthetic) data.

Can we recover the mixing weights (target source to data channels)?

* mcca\_demo3.m

shows how. Note that the estimate is unreliable at low SNRs

What happens if there is more than one shared source?

* mcca\_demo4.m

illustrates this situation. Fig 1 shows the profile of variance per SC (two SCs have high values). Fig 2 shows the original sources (left) and the recovered components (right). The recovered components (first 2 SCs) span a *subspace* containing the original sources. The subspace is well isolated, but there is no one-to-one mapping between sources and recovered components.

What happens if each source is shared among several data matrices, but not all?

* mcca\_demo10.m

shows the result of applying MCCA to data containing 20 sinusoidal target sources shared by data matrices two-by-two. Each data matrix is a mixture of 4 sinusoidal targets (each shared by one other data matrix) and 6 Gaussian noise sources, mixed via a random matrix. MCCA successfully finds all 20 target sources. Fig 1 shows the profile of variance across SCs (the first 20 SCs have elevated variance), and Fig 2 (top) shows the time courses of the first 20 SCs. Note that all 20 targets are recovered, despite the fact that each dataset has only 10 channels.

What happens if the data are rank-deficient (for example ICA has been used to project out artifacts)?

* mcca\_demo5.m

deals with this situation by reducing dimensionality (low-variance PCs are discarded in the initial PCA step).

What happens if the noise is full rank (i.e. signal and noise are not linearly separable)?

* mcca\_demo6.m

looks at this situation with several profiles of correlation within the noise. Fig 1 (left) plots the eigenspectra of different noise sets, from flat (all noise dimensions have equal variance) to steeply decreasing (some dimensions have much lower variance). Fig 1 (right) shows the absolute value of the correlation coefficient between the target source and the recovered signal (first SC) as a function of SNR. If the noise eigenspectrum is steep, MCCA can recover sources at lower SNR than when the noise eigenspectrum is flat.

What happens if the noise is spectrally colored (as for real EEG noise)?

* mcca\_demo7.m

shows the result of MCCA applied to low-pass noise in the absence of any target. The noise was obtained by integrating Gaussian white noise (cumsum(randn())), which gives a 1/f^2 spectrum (similar to a random walk). Its spectrum is shown in Fig 1 top left. All dimensions of the noise have this same spectrum, and data channels and matrices are independent from one another. Despite this, MCCA discovers 'shared' dimensions that tend to have a low frequency content, as evident from the spectra of all SCs (Fig 1 bottom left) and waveforms of individual SCs.

* mcca\_demo8.m

shows a similar result for narrowband filtered noise. These 'shared' components are of course spurious. MCCA has a strong bias towards low-frequency or narrowband components within the data.

EEG signals are typically superimposed on similar slow drift components.

* mcca\_demo9.m

shows the result of applying processing similar to mcca\_demo7.m to real EEG data. Slow drifts in the EEG (Fig 1) are enhanced by MCCA (Fig 2). The first few SCs tend to be quasi-sinusoidal. SCs tend to be ordered by increasing frequency content (Fig. 3).

How to test whether shared patterns observed are real or due to spectral coloring of noise? One way is to apply the data to surrogate data obtained by randomizing the phase of all channels of all data matrices. These surrogate data have the same power spectrum (and thus same autocorrelation) as the original data, but any genuine shared temporal pattern should be washed out due to the phase scrambling.

* mcca\_demo12.m

applies phase randomization to the low-pass noise dataset of mcca\_demo7. The clear shape of the first SC is unlikely to be a genuine shared component, because each randomization produces a similarly-shaped component (Fig 2) (albeit with a different temporal offset). In contrast,

* mcca\_demo13.m

applies phase randomization to a dataset containing a real target embedded in low-pass noise. Applying MCCA to the unscrambled data reveals the target (Fig1) but applying the same analysis to scrambled data fails to find it (Fig2), indicating that the pattern found by MCCA in the original data was real.

When to use SCs, when to use CCs? SCs are a transform of all the data, whereas each set of CCs is a transform of just one data matrix. SCs are orthogonal, CCs are not (in general). SCs are useful to derive a 'summary' of all the data matrices, whereas CCs can be used to denoise each individual data matrix, with no contribution from other data matrices. This can be useful to preserve patterns specific to each data matrix (for example latencies that differ between subjects).

* mcca\_demo15.m

applies MCCA to a dataset consisting of 9 data matrices containing 3 targets. Each target is shared among 3 data matrices, while each data matrix is affected by only one target. The SCs each contain a mixture of the targets (i.e. no SC actually matches a target) (Fig 1). In contrast, each set of CCs contains only one target (Fig. 2). This example shows how MCCA can isolate targets that are only partially shared across data matrices.

What happens if the target is not quite the same between data matrices? It turns out that the answer depends on the nature of the background noise.

* mcca\_demo17.m

shows the result of applying MCCA to a dataset in which a shared target is mixed with background noise into 10 data matrices. For each data matrix, a data matrix-specific mismatch signal (Gaussian noise) is added to the target with an amplitude that is varied as a parameter. The background noise is either white, or low pass (1/f^2, implemented as cumsum(randn())). For a white noise background, MCCA succeeds in recovering the target (Fig 1). For a colored-noise background, MCCA may fail if the amplitude of the data matrix-specific mismatch is large (Fig 2). This may be understood as a competition between the (imperfectly shared) target and (imperfectly shared) components spuriously correlated between data matrices.

Interestingly, the threshold amplitude is lower if the data have more channels (Fig 3), suggesting a role for the dimensionality of the data.

Is there benefit in applying dimensionality reduction? The answer depends on several parameters, including the signal-to-noise ratio, the spectral shape and variance profile (cf. mcca\_demo6) of the background noise, and the number of dimensions retained.

* mcca\_demo18.m

illustrates this. For a wide range of parameter values, applying dimensionality reduction to each data matrix before MCCA improves its performance (Figs 1-3). Typically, there is a range of number of dimensions retained ("sweet spot") within which MCCA succeeds.

How does MCCA fare on real brain data? We use a dataset of 64-channel EEG responses to a repeated auditory stimulus (short 1kHz tone), measured in 25 subjects.

* mcca\_demo20.m

applies PCA to each data matrix, selects a subset of PCs to reduce dimensionality, and then applies MCCA. The first few SCs turn out to be quite replicable over trials (Fig 1), suggesting that MCCA has successfully isolated cortical activity time-locked to sound. The SCs are mutually orthogonal, but there is great similarity between their trial-averaged time courses. We can remove redundancy and summarize these time courses by applying PCA to the trial-averaged time courses (Fig 2).

The same dataset (as used in mcca\_demo20) can be analyzed subject-by subject based the assumption that each stimulus elicits the same brain response.

* mcca\_demo16.m

uses DSS (aka JD or CSP) to find spatial filters that optimize the repeatable response.

For each subject, the first DSS component is the most reproducible linear combination of channels (Fig 1). In addition to this first component, there are others that contribute to the repeatable response (Fig 2). Together they span a signal subspace that reflects the activity of multiple brain sources (with distinct temporal and spatial signatures).

To the extent that stimulus-excited brain sources are similar across subjects, we expect MCCA to be able to isolate them. The first 5 DSS components for each subject were concatenated across subjects and submitted to MCCA analysis. The outcome (Figs 4 and 5) suggests that the space of all sources across all subjects is spanned by about 5-6 components. These may reflect both multiple brain sources within each subject, and between-subject differences in time course of activation of these brain sources.

Might the results found by mcca\_demo16 be spurious, reflecting overfitting?

* mcca\_demo16b.m

applies the analysis to the same data with phase-scrambling of each channel of each subject. Variance scores for intact data appear to be well outside the distribution for phase-scrambled data, for the first 5-6 SCs (Fig 1). The same analysis was performed on reduced-dimensionality data (20 PCs from PCA applied to each subject's preprocessed EEG). Here, variance scores for intact data emerge from the distribution of phase-scrambled data for only ~4 SCs, suggesting that dimensionality reduction might have removed genuine sources.

How important is preprocessing of EEG data?

* mcca\_demo21.m

shows the result of applying MCCA to data with and without highpass filtering or detrending. In the absence of any filtering (Fig 1) processing fails because the slow trends common to all subjects are selected by MCCA instead of interesting patterns. It succeeds with high-pass filtering at 1 Hz (Fig 5), or 10th-order polynomial detrending of each epoch (Fig 4) but fails for high-pass filtering at 0.1 Hz. Preprocessing is important to avoid locking on to trivial drift patterns.

Can MCCA and DSS be used to improve each other's performance, for example by using one as a denoising or dimensionality reduction tool for the other, and vice-versa?

* mcca\_demo22.m

shows how this might be done. Alternating between DSS and MCCA increases the separation between the MCCA variance score for intact data relative to the distribution for phase-scrambled data, suggesting that the strategy is effective.