Introduction to BCILAB A MATLAB Toolbox and EEGLAB Plugin

Christian A. Kothe Swartz Center for Computational Neuroscience, UCSD

EEGLAB Overview



What is EEGLAB?

- Large open-source toolbox for EEG analysis (70k lines, 90k d/loads, 5000-9000 users on discussion list)
- Neuroscience focus & features (ICA, 3d source localization, statistics, multi-subject analysis, graphics)
- Developed by Arno Delorme and Scott Makeig (et al.) under NIH funding
- 20+ plugins (NFT, SIFT, BCILAB, MPT, ...)
- Currently being extended for real-time experimentation (MoBILAB, ERICA platform)





EEGLAB articles



Delorme, A., Makeig, S. (2004) EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics including independent component analysis. *Journal of Neuroscience Methods*, 134(1), 9-21.

Makeig, S., Debener, S., Onton, J., Delorme, A. (2004) Mining event related dynamics. *Trends in cognitive Neuroscience*, 8(5), 204-210.

Delorme, A., Kothe, C., Bigdely, N., Vankov, A., Oostenveld, R., Makeig, S. Matlab Tools for BCI Research? In "human-computer interaction and brain-computer interfaces". Editors : Tan, D. and Nijholt, A. To appear in 2010. Springer Publishing.

Delorme, A., Mullen, T., Kothe, C., Bigdely-Shamlo, N., Akalin, Z., Vankov, A., Makeig, S. EEGLAB, MPT, NetSIFT, NFT, BCILAB, and ERICA: New tools for advanced EEG/MEG processing. Computational Intelligence, accepted.

Delorme, A., Makeig, S. Open Source Programming for Interpreted Language: Graphic Interface and Macro Bridging Interface. IEEE International Conference on Signal Image Technology and Internet Based Systems. In press.

NFT: Neuroelectromagnetic Forward Head Modeling Toolbox



http://sccn.ucsd.edu/nft





BCILAB Overview



What is BCILAB?



http://sccn.ucsd.edu/wiki/BCILAB



Idea & Purpose

- Like EEGLAB, but for BCI (and/or cognitive state assessment)
 - Seeding a community
 - Strengthening links between BCI and Neuroscience
- SCCN's in-house tool for BCI problems
 - Main focus: Advanced cognitive monitoring
 - Part of a large US research program (CaN CTA)
 - Funded by ARL (and ONR, Swartz Foundation, ...)





Research Directions

- HCI: User-state monitoring, intelligent assistive systems
- **Neuroscience**: Brain feedback experiments
- **Clinical**: Communication and control devices for the severely disabled
- Entertainment: Computer game controllers





Research Directions

• Neuroscience: also, *decoding models* of brain dynamics (exploratory research)





BCILAB's Niche

- State of the art
- Largest collection of machine learning & signal processing components in any open-source BCI package
 - Many standard components (CSP, LDA, SVM, ...)
 - Many modern components (SBL, SSA, AMICA, HKL, DPGMM, LR-DAL, ...)
 - Some novel components (OSR, RSSD, SSB, ...)
- Next-generation framework
 - Fully probabilistic
 - Model inference from data corpora*
 - Anatomical priors, other neuroscience-aware features
 - Processing of parallel streams

(*: not yet in the current release)



(Intangible) Aims

- Low entry barrier
 - Developers: Simple plugin framework, low overhead
 - *Experimenters*: User-friendliness, GUI, canned approaches
- Low usage friction
 - Flexible, unobstructive
 - Simple things easy to achieve, complex things possible
- Efficiency
 - No redundant computations (caching, ...)
 - Parallel computation
 - Capable scripting (batch analysis, parameter search, ...)
 - Automation

Theory, Terminology and BCILAB Equivalents



Signals

- We measure one or more (multi-channel, fixed-rate) *signals* of a person
 - EEG, ECoG, MEG, ...
 - EMG, EOG, Gaze, MoCap, ...

white months and the property and the property and the property of the propert

man water and a second and a



Signal Representations





•

Bundle n



Data Sources

- Signals in BCILAB originate from *device plugins*
- Currently: BioSemi, TCP, OSC, BCI2000, DataRiver, playback, etc.
- More planned (e.g., BrainProducts, ANT, g.Tec, Emotiv, ...)
- Quite easy to extend (typically a few 10s of LoC)



Basic Framework

- The goal is to build an oracle that consumes these signals and can answer (pre-defined) queries about cognitive state of the person
- BCIs fit into that framework





Basic Framework

- In BCILAB, outputs may be point estimates (scalar, vector) or probability distributions (discrete / continuous)
 - usually discrete prob. distributions





Basic Framework

• A BCI is specified/described by a "BCI model"





- All BCI models contain a mapping f(X) that maps a limited-length signal segment $X \in \mathbb{R}^{C \times T}$ onto the output y
 - In BCILAB called the model's "prediction function"
 - May also accept segments from multiple signals





- Note: the prediction function involves a functional form and possibly some fixed parameters
 - Functional form reflects a relationship between measurements and cognitive state (inverse for some assumed generative mechanism)





- May also apply *signal processing methods* (here called *filters*) to the input signals (e.g., for computational efficiency or to leverage tools)
 - receive a signal and produce a transformed signal
 - online-capable and possibly adaptive / stateful
 - represented as plugins in BCILAB, more than 40 methods built in





Major Filter Types

- Spatial filters (channel selection, surface Laplacian, ICA, CSP*, sparse recovery, ...)
- Spectral filters (FIR, IIR, FFT)
- Epoch-based filters (windowing, Wavelet transform, Fourier transform, STFT, ...)
- Miscellaneous (resampling, dipole fitting, ...)



- Putting all together, a BCI model in BCILAB contains a *filter graph* and a prediction function
- Provides enough flexibility for most BCI designs





Online Data Flow

• In BCILAB, the filter graph receives all input samples, but the prediction function may be called on demand





Online Data Flow

- In BCILAB, the filter graph receives all input samples, but the prediction function may be called on demand
 - In most current BCIs, the prediction function consists of a dedicated "feature extraction" step and "prediction" step





One Problem

- for most BCI questions and implementations, the parameters leading to best accuracy are *a priori* unknown!
 - Depend on hard-to-measure factors (e.g., brain functional map)
 - Depend on expensive-to-measure factors (e.g., brain folding)
 - Depend on highly variable factors
 (e.g., sensor placement, subject state)
 - Different for every person, task, montage, etc.



One Problem

• Example per-channel parameters across four subjects:



(image: Blankertz et al. 2007)



Model Calibration

• Need *calibration / training data* to estimate parameters from and a separate *calibration step*





Model Calibration

 In theory many possibilities (e.g. MR scanner data + Beamforming)



Model Calibration

- In theory many possibilities (e.g. MR scanner data + Beamforming)
- Most successful way (so far): utilize data where both the BCI input (e.g. EEG) and desired output (cognitive state) is known – in BCILAB called "calibration recording" – and adapt BCI parameters using machine learning algorithms Model machine learning Calibration recording



Calibration Recording

- Standard psychological experiment
 - continuous EEG (or other)
 - multiple trials/blocks (capturing variation)
 - randomized (eliminating confounds)



Calibration Recording

- Standard psychological experiment
 - continuous EEG (or other)
 - multiple trials/blocks (capturing variation)
 - randomized (eliminating confounds)
 - event markers to encode cognitive state conditions of interest, e.g., stimuli/responses (called *"target markers"* in BCILAB)





Machine Learning

- Large field with 100s of algorithms
- Most methods conform to a common interface of a *training function* and a *prediction function*
 - training function accepts a matrix of feature vectors (samples) $X \in \mathbb{R}^{N \times F}$ and target values $y \in \mathbb{R}^{N \times D}$ with F the # of features per sample, N the # of samples, D the dimensionality of the target space (usually D=1); the output is a *model* with parameters θ
 - *prediction function* accepts a matrix of feature vectors
 X, model parameters *θ* and produces estimates of the corresponding target values *y*


Machine Learning

- BCILAB has a plugin framework for machine learning (with >60 built-in algorithms)
- Supports some additional formats for X, e.g., matrixvalued features and y (e.g., common distributions)
- Training function also accepts additional parameters





Model Calibration, cont.

• In BCILAB, typically one trial segment (sample) is extracted for every target marker in the calibration recording





Feature Extraction

- Problem: Standard machine learning methods often do not work very well when applied to raw signal segments X of the calibration recording
 - too high-dimensional (too many parameters to fit)
 - too complex structure to be captured (too much modeling freedom)
 - (but note: different story for custom methods)

1000s of degrees of freedom





Feature Extraction

- Solution: Introduce additional mapping (called *"feature extraction")* from raw signal segments onto feature vectors
 - output is often of lower dimensionality
 - hopefully better distributed in the feature space (easy to handle for machine learning)



Model Calibration, cont.

• With feature extraction, the analysis process is as follows:





BCI Paradigms

- BCI paradigms are the coarsest plugin type in BCILAB and tie all parts of a BCI approach together
 - *calibrate function*: accepts a calibration recording (with markers), additional parameters, and produces a BCI model; may utilize machine learning, signal processing and/or other methods
 - *prediction function*: serves as the prediction function for the resulting BCI model
 - optionally additional code, e.g., for feature extraction and feature adaptation (feature learning and/or feature selection)



BCI Paradigms

- BCI paradigms are the coarsest plugin type in BCILAB and tie all parts of a BCI approach together
- Note: Multiple approaches can be realized with a single paradigm (using different parameters)





BCI Paradigm Plugins

- For event-related potentials
 - Time-window averages (explained next)
 - Wavelet features
 - First-order DAL
- For oscillatory processes
 - Common Spatial Patterns, Regularized CSPs, Spectrally weighted CSP, ...
 - Channel-based approaches
 - ICA-based approaches (OSR, RSSD, …)
 - Second-order DAL

Concrete Approach for ERPs



Example Approach for ERPs

- Suppose a calibrate function with steps:
 - 1. Apply an IIR band-pass filter to raw data (0.5 15 Hz)
 - 2. Extract 0.8 s segments around stimuli in recording
 - 3. Extract features and run machine learning



Example Approach for ERPs

- Suppose a calibrate function with steps:
 - 1. Apply an IIR band-pass filter to raw data (0.5 15 Hz)
 - 2. Extract 0.8 s segments around stimuli in recording
 - 3. Extract features and run machine learning
- Applied to a recording with 100 stimuli of type A and 100 stimuli of type B





Resulting filtered segments









Extracting Features





For each trial segment, calculate signal mean in three time windows (gives 3-dim feature vectors per trial)





Using Machine Learning

• The feature vectors are passed on to a machine learning function (e.g., Linear Discriminant Analysis)





LDA In a Nutshell

• Given trial segments x_k (in vector form) in \mathcal{C}_1 and \mathcal{C}_2 ,

$$\boldsymbol{\mu}_i = \frac{1}{|\mathcal{C}_i|} \sum_{k \in \mathcal{C}_i} \boldsymbol{x}_k, \qquad \Sigma_i = \sum_{k \in \mathcal{C}_i} (\boldsymbol{x}_k - \boldsymbol{\mu}_i) (\boldsymbol{x}_k - \boldsymbol{\mu}_i)^{\mathsf{T}}$$

 $\boldsymbol{\theta} = (\Sigma_1 + \Sigma_2)^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1), \qquad \mathbf{b} = \boldsymbol{\theta}^{\mathsf{T}} (\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)/2$





LDA In a Nutshell

• Given trial segments x_k (in vector form) in \mathcal{C}_1 and \mathcal{C}_2 ,

$$\boldsymbol{\mu}_{i} = \frac{1}{|\mathcal{C}_{i}|} \sum_{k \in \mathcal{C}_{i}} \boldsymbol{x}_{k}, \qquad \boldsymbol{\Sigma}_{i} = \sum_{k \in \mathcal{C}_{i}} (\boldsymbol{x}_{k} - \boldsymbol{\mu}_{i}) (\boldsymbol{x}_{k} - \boldsymbol{\mu}_{i})^{\mathsf{T}}$$
$$\boldsymbol{\theta} = (\boldsymbol{\Sigma}_{1} + \boldsymbol{\Sigma}_{2})^{-1} (\boldsymbol{\mu}_{2} - \boldsymbol{\mu}_{1}), \qquad \mathbf{b} = \boldsymbol{\theta}^{\mathsf{T}} (\boldsymbol{\mu}_{1} + \boldsymbol{\mu}_{2})/2$$

- **Caveat**: θ often high-dimensional but only few trials available
- Can use a regularized estimator instead, here using shrinkage; instead of Σ_i, we use Σ̃_i above:

$$\tilde{\Sigma}_i = (1 - \lambda)\Sigma_i + \lambda I$$



Machine Learning Plugins

- Generative Models (LDA, RLDA, QDA, GMMs)
- Discriminative Models (SVMs, RVMs, GLMs, HKL, ...)
- Custom Frameworks (convex optimization, graphical models, ...)



Calibration: Summary

- Basic calibration in BCILAB typically involves:
 - Filtering the data (possibly adapting filters)
 - Extracting trial segments and features (if necessary)
 - Applying a machine learning function
 - Specifying the model structure (filter graph, prediction function, parameters)



Visualizing θ

- Linear model weights can be visualized as a (color-coded) value per time window and channel
- Below: 6 windows, 21 channels, ERP task

 Window1 (0.25s to 0.3s)
 Window2 (0.3s to 0.35s)
 Window3 (0.35s to 0.4s)

 Image: Window4 (0.4s to 0.45s)
 Image: Window5 (0.45s to 0.5s)
 Window6 (0.5s to 0.55s)







Evaluating Model Performance



Offline Evaluation

- Given calibration data
- Estimate model parameters (spatial filters, statistics)
- Apply the model to new data (online / single-trial)
- Optionally: compare outputs with known state, compute loss statistics for the model / approach (e.g., misclassification rate)





Offline Evaluation

Evaluation of computational approaches on a single data set?



Calibration recording



Offline Evaluation

- Evaluation of computational approaches on a single data set?
 - Can not test on the training data! (always on separate data)
 - Instead can split data set repeatedly into training/test blocks systematically, a.k.a. cross-validation





Resolving Free Parameters

- Can be done using cross-validation in a grid search (try all values of free parameters)
- Caveat: Resulting "optimal" numbers are non-reportable (cherry-picked!)





Resolving Free Parameters

- Can be done using cross-validation in a grid search (try all values of free parameters)
- Caveat: Resulting "optimal" numbers are non-reportable (cherry-picked!)
- But may test resulting best model on separate data





Resolving Free Parameters

- Can be done using cross-validation in a grid search (try all values of free parameters)
- Caveat: Resulting "optimal" numbers are non-reportable (cherry-picked!)
- But may test resulting best model on separate data
- **Or** run grid search *within* an outer cross-validation ("nested cross-validation")



Next: Basic GUI Tour

Startup

-1	MATLAB 7.6.0 (R2008a)							
Fi	le E	Edit	Debug	Parallel	Desktop	Window	Help	
F								
								_
Þ	>> c	cd (C:/DEV	/EL/Mat	tlab/bo	ilab-0	0.9-stable/; startup	
Ľ								
4	Sta	art R	Ready				OVE	

Toolbox GUI

MATLAB 7.6.0 (R2008a)							
File Edit Debug Parallel Desktop Window Help							
Data Source Offline Analysis Online Analysis Settings Help							
results are in C:\DEVEL\Matlab\bcilab-0.9-stable\\userdata							
cache is in C:\tmp\bcilab_cache (location_1)							
temp is in C:\tmp\bCllab_temp							
Welcome to the BCILAB toolbox!							
>>							
A Start	OVR //						

A BCILAB 0.9							
Data Source	Offline Analysis	Online Analysis	Settings	Help			
				BCI Paradigms			
				Filters			
				Machine Learning	J		
				Scripting	•		
				Plugin authoring	•		
				About			
				Save bug report.			

🕐 Help								
File Edit View Go Favorites Desktop Window Help								
Help Navigator X	🗭 🔶 😂 🖊							
Search for: Go	Title: M-File Help: code/filters	T						
Example: "plot tools" OR plot* tools Contents Index Search Results Demos	code/filters	Default Topics						
🖽 🤣 Release Notes 📃								
	Contents of filters:							
🗉 🧼 Aerospace Toolbox	flt clean channels	- Remove channels with abnormal data from a continuous data set. Currently offline o						
🗄 🧼 Bioinformatics Toolbox	flt clean spikes	- Set outliers in data to zero.						
📗 🖶 🧼 Communications Toolbox 🛛 🛁	flt clean windows	- Remove periods of abnormal data from continuous data.						
🗄 🤣 Control System Toolbox	flt envelope	- Compute the signal envelope for a continuous data set. Non-causal.						
🕀 🧼 Curve Fitting Toolbox	flt furier	- Filter a continuous data set by a digital FIR filter.						
🗄 🤣 Database Toolbox	flt ica	- Annotate the Signal with a spatial decomposition into independent components (usin						
🕀 🌛 Datafeed Toolbox	flt iir	- Filter a continuous data set by a digital IIR lowpass/highpass/bandpass/bandstop f						
Embedded MATLAB	fit laplace	- Applies a simple Hjorth-style surface laplacian filter.						
Eilter Design Toolbox	flt project	- The default extensible preprocessing pipeline for most but paradigms. - Snatially project the given data set, e.g. to apply an IC decomposition						
Eilter Design HDL Coder	flt reconstruct	- Reconstruct the given data in a new (possibly overcomplete) basis.						
Einancial Toolbox	<u>flt reref</u>	- Re-references the data to a new (set of) channel(s) or the average of all channels						
Einancial Polibox	flt resample	- Changes the sampling rate of a given data set.						
	flt selchans	- Subtract a baseline from an data set, computed over the given baseline window.						
Fixed-Income Toolbox	flt spectrum	- Select a frequency portion of the data in an epoched data set.						
	flt standardize	- Standardize a continuous EEG set causally.						
H Spice Logic Loolbox	flt window	- Select a time portion of the data in an epoched data set.						
		Þ						





http://sccn.ucsd.edu/wiki/BCILAB

Use Case A

- You just recorded pilot data for some new study
- The idea is to try to estimate a certain aspect of cognitive state
- The question is what method works best, and what accuracies can be achieved

Use Case A

- Scenario: Subject is instructed to imagine a hand movement, either left hand or right hand (standard BCI case)
- Task: Estimate, from raw data, which hand movement was imagined
- Experimental data: EEG, 32 channels, 2 sessions (each ~30 min.)
Experimental task

- 160 trials
- Randomized Instruction: L or R
- Displayed for 3s, followed by blank screen for 3.5s
- Sample:



Loading the Data

	📣 BCILAB 0	.9					
	Data Source	Offline Analysis	Online Ar	alysis	Settings	Help	
Ĩ	Load recor	rding(s)	Ctrl+L				
	Load study	ý		-			
	Define ma	rker transform	Ctrl+D				
	Workspace	e	•				

Loading the Data

📣 BCILAB 0	.9				
Data Source	Offline Analysis	Online Analysis	Settings	Help	

-	Select datas	et(s) to load				×
	Look in:	퉬 test			## # •	
	A - a	Name 🔺		→ Date modified →	Туре	▼ Size
	2	errors.fdt		11/19/2010 3:59	FDT File	8,437 K
R	lecent Places	errors.set		11/19/2010 3:59	SET File	555 K
		errors2.fdt		11/19/2010 3:59	FDT File	6,594 K
		errors2.set		11/19/2010 3:59	SET File	351 K
	Desktop	imag.fdt		11/21/2010 10:0	FDT File	15,744 K
	<u></u>	imag.set		11/21/2010 10:0	SET File	129 K
		imag2.fdt		11/21/2010 10:0	FDT File	15,835 K
	Libraries	imag2.set		11/21/2010 10:0	SET File	127 K
	-					
	- 1					
	Computer					
II.	(
	Network					
		•				
		File name:	imag.set		–	Open
		Files of type:	any supported file		-	Cancel
_			Terry copported no			

Loading the Data

📣 BCILAB 0.9				_ 🗆 🗙
Data Source Offline An	alysis Online Analysis	Settings	Help	

ł	Load source data
	Optionally load data in reduced form
	Channel index subset
	Sample range subset
	Time range subset
	— Optionally add fields, if missing (raw files only)————
	Sampling rate
	Saue data in workenace as
	Add markers Cancel OK

 An approach addresses both parts of the BCI problem: Mapping from observed signals to predictions, and learning the unknown parameters

📣 BCILAB 0	.9			_
Data Source	Offline Analysis Online Analys	is Setting	gs Help	
<u> </u>	New approach	Ctrl+N		
L	Modify approach	Ctrl+M	μ	
	Review/edit approach	Ctrl+R		
	Save approach	Ctrl+S		
	Train new model	Ctrl+T		
	Apply model to data	Ctrl+A		
	Visualize model	Ctrl+V		
	Investigate results	Ctrl+I		
	Transform data by model	Ctrl+P		

 You never start completely from scratch, but on the basis of what is known to work



• Some of these work best for oscillatory processes, others for ERP-like features, etc.



📣 BCILAB 0.	.9				
Data Source	Offline Analysis	Online Analysis	Settings	Help	

📣 Define a new approach	_ 🗆 X
— Select approach ————	Also see help
Common Spatial Pattern (para_	_csp) text
Description	
Standard paradigm for oscillatory (CSP) algorithm. Result = para_cs	v processes via the Common Spatial Patterns sp(Input-Data, Operation-Mode, Options)
The CSP paradigm is based on th (BBCI) [1], more comprehensively (sensori-)motor imagery. The fea are Event-Related Synchronizatio (sensori-)motor cortex, but the pa CSP was originallyintroduced in [tures exploited by this paradigm in its original form onand Desynchronization [3] localized in the aradigm is not restricted to these applications. 5] and first applied to EEG in [6].
Due to its simplicity, speed and re paradigm for oscillatory processe estimate of whether the data con para_bandpower, CSP uses log-	elative robustness, CSP is the bread-and-butter es, and if nothing else, can be usedto get a quick itains information of interest or not. Like variance features over a singlenon-adapted
Help	Cancel Refine OK

📣 BCILAB 0	.9				
Data Source	Offline Analysis	Online Analysis	Settings	Help	

•)	Define a new approach		
Г	- Select approach		
	Common Spatial Pattern (para_csp)	•	
	- Description		
	Standard paradigm for oscillatory processes via the Common Spatial Patterns (CSP) algorithm. Result = para_csp(Input-Data, Operation-Mode, Options)	-	
	The CSP paradigm is based on the design of the Berlin Brain-Computer Interface (BBCI) [1], more comprehensively described in [2], which is mainly controlled by (sensori-)motor imagery. The features exploited by this paradigm in its original for are Event-Related Synchronizationand Desynchronization [3] localized in the (sensori-)motor cortex, but the paradigm is not restricted to these applications.	; prm	۸da
	CSP was originally introduced in [5] and first applied to EEG in [6]. Due to its simplicity, speed and relative robustness, CSP is the bread-and-butter paradigm for oscillatory processes, and if nothing else, can be used to get a qui estimate of whether the data contains information of interest or not. Like	24	to y
	para_bandpower, CSP uses log-variance features over a singlenon-adapted	_	
	Help Cancel Refine (ж	

Adapt the template to your experiment

Configuring an Approach

Key properties can be configured in this dialog

	BCILAB 0.9 Data Source Offline Analysis Online Analysis Settings Help	
	Configure approach	
	New sampling rate of the data	100
	Epoch time window relative to the events Event/marker types for which epochs shall be extracted	()
	Frequency-domain selection	[7 30]
- · ·	Number of CSP patterns (times two)	3
adapted	h properties usually need to be Machine learning function	Ida
	Help	Cancel Ok

Configuring an Approach





Also, target marker types in the data have to be specified

Reviewing/Editing an Approach

- The next panel allows to edit all properties of the approach
- Filter stages can be added and configured
- Feature extraction can be configured
- Machine learning components can be selected and configured
- For now, nothing to do

Signal Processing	
 SignalProcessing 	
FilterOrdering	
SamplingRate	100
ChannelSelection	
Rereferencing	
ICA	
SurfaceLaplacian	
FIRFilter	
Projection	
IIRFilter	
Standardization	
SparseReconstruction	
EpochExtraction	
TimeWindow	[0.5 3.5]
EventTypes	5 1 []
BaselineRemoval	
WindowSelection	
SpectralTransform	
 SpectralSelection 	
FrequencySpecification	[7 30]
Feature Extraction	
FeatureExtraction	
PatternPairs	3
PluginFunctions	
FeatureAdaptor	para_csp::@csp_train
FeatureExtractor	para_csp::@csp_predict
FeatureAdaptorNeedsVoting	
FeatureVisualizer	para_csp::@csp_visualize
Machine Learning	
Machinel earning	Y
(Name)	

Saving the Approach



Learning a Predictive Model

• Put the method to the test...

📣 BCILAB 0	.9				
Data Source	Offline Analysis	Online Analysi	is Settings	; Help	
	New approact	n	Ctrl+N		
	Modify approa	ach	Ctrl+M		
	Review/edit a	pproach	Ctrl+R		
	Save approac	h	Ctrl+S	7	
	Train new mo	del	Ctrl+T		
L	Apply model t	o data	Ctrl+A	-	
	Visualize mode	el	Ctrl+V		
	Investigate re	sults	Ctrl+I		
	Transform dat	a by model	Ctrl+P		

Approach and data to use

data to use	Calibrate a model			
	Selected approach	lastapproach ("Imagined Mo	ovements vi 💌	
	Calibration data source	lastdata ("imag.set")	•	
- ВСТІ	Parameter Search			1
Data So	Loss/Performance Metric	Automatically chosen	-	
	Cross-validation folds	5		ľ
	Spacing around test trials	5		
			Defines	the performance
	Performance estimates		metric	
~	Compute performance esti	imates	metric	
Check to get	Cross-validation folds	10 🗲	- Smalle	r number: faster,
performance	Spacing around test trials	5	bu	t lower quality
estimates	Computing resources			estimates
	Run on a computer cluster			
	Node pool	(use current co	onfig)	
	Save model in workspace as	lastmodel		
	Save stats in workspace as	laststats		
	Help	Cano	cel OK	

Waiting for Results

MATLAB 7.6.0 (R2008a)	IX
File Edit Debug Parallel Desktop Window Help	
creating epochs	
pop epoch():160 epochs selected	
Epoching	
pop epoch():160 epochs generated	
pop epoch(): checking epochs for data discontinuity	
loading C:\DEVEL\Matlab\bcilab-0.9-stable\userdata\test\imag.set	
pop loadset(): loading file C:\DEVEL\Matlab\bcilab-0.9-stable\userdata\test\imag.se	<u>.</u>
Reading float file 'C:\DEVEL\Matlab\bcilab-0.9-stable\userdata\test\imag.fdt'	
The loaded EEGLAB set is lacking an online expression; assuming it contains unfilte	2:
If it contains filtered data, however, BCI models derived from it will likely not k	
creating epochs	
pop_epoch():160 epochs selected	
Epoching	
pop_epoch():160 epochs generated	
pop_epoch(): checking epochs for data discontinuity	
>>	•
<mark>∢ Start</mark> ØV	R

Reviewing Results



Reviewing Results

- 11% error rate is quite good for imagined movements; mean across studies & methods is probably closer to 25%
- chance level is here 50% (keep that in mind when evaluating)
- You may get multiple outputs (e.g., false positives, true positives, which show up in the table), depending on loss measure

Visualizing Model Properties

A BCILAB 0	.9			_ 🗆 🗙
Data Source	Offline Analysis Online Analys	is Setting	gs Help	
	New approach	Ctrl+N		
	Modify approach	Ctrl+M		
	Review/edit approach	Ctrl+R		
	Save approach	Ctrl+S		
	Train new model	Ctrl+T		
Г	Apply model to data	Ctrl+A		
	Visualize model	Ctrl+V		
L	Investigate results	Ctrl+I		
	Transform data by model	Ctrl+P		

Visualizing Model Properties



Apply Model to 2nd Session



Apply Model to 2nd Session

📣 BCILAB 0	.9			_ 🗆 🗙
Data Source	Offline Analysis Online Analys	is Setting	gs Help	
ų.	New approach	Ctrl+N		
	Modify approach	Ctrl+M		
	Review/edit approach	Ctrl+R		
	Save approach	Ctrl+S		
	Train new model	Gtrl+T		
	Apply model to data	Ctrl+A		
L	Visualize model	etrl+V		
	Investigate results	Ctrl+I		
	Transform data by model	Ctrl+P		

Apply Model to 2nd Session

📣 BCILAB 0.	9				
Data Source	Offline Analysis	Online Analysis	Settings	Help	

Apply predictive model to dat	a set
Source data set for prediction	lastdata ("imag2.set")
Predictive model to use	lastmodel
Loss/performance metric	Automatically chosen
Save results in workspace as	lastresults
Help	Cancel OK

Reviewing Results

📣 Review Results	<u> </u>
Data Summary	
Error rate : 0.09 +/- 0.00 (N=1)	
- Data Details	
Error rate	
1	0.0938
Help Explore Export	Save OK

OR: Apply Model Online

- (if you have a subject sitting next to you)
- Here: use a simulated data source (playing back the 2nd session)

📣 BCILAB 0	.9				
Data Source	Offline Analysis	Online Analysis	Settings	Help	
		Process data	within	→	
		Receive input	t from	•	DataRiver
		Provide output	ut to	•	BrainVision Recorder
		Clear all onlin	e processir	ng 🗖	тср
					Simulated (dataset playback)
				-	

Online Application

This adds a data feed process in the background



🥠 gui_receivedataset		<u>- </u>
Dataset to replay	lastdata ("imag2	2.s 🔻
Update frequency	25	
Save stream in workspace as	laststream	
Cancel	Cancel	ОК

Online Application

📣 BCILAB 0	.9				_ 🗆 🗙
Data Source	Offline Analysis	Online Analysis	Settings	Help	
		Process data	within	►	
		Receive input	t from		
		Provide output	ut to	Þ	MATLAB visualization
		Clear all onlin	e processi	ng	DataRiver
					TCP

Online Application

This adds a real-time inference process in the background

BCILAB 0.	9		
ata Source	Offline Analysis Online Analysis Setti	ings Help	
	gui_producevisualization		25 Hz if your
	Predictive model to use	lastmodel 🗾	computer is fast
	Real-time stream to read	laststream 💽	enough
	Update frequency	10	
	Output format	distribution 💌	Presented via this display command
	Visualization function	bar(y)	
	Save predictor in workspace as	lastpredictor	
	Help	Cancel OK	

Real-time Output





Real-time Output

- If you have more classes, you get more bars
- You can also remap to other parameters (e.g. expected value)
- Note: the simple graphics command always renders into the current window

Using a More Ambitious Approach

A BCILAB 0.9								
Data Source	Offline Analysis	Online Analysis	Settings	Help				
Data Source Define a Select a Spect log-B Comr Dual- High- a Low- Data- Indep Multik Mulip Spect T Vind a (From 1 t Imagi c (From 1 t Imagi c Second Band BCIS0 Error Band BCIS0 Error	Offline Analysis a new approach pproach rally Weighted CSF andpower (para_b non Spatial Pattern Augmented Lagran Frequency DAL (p Frequency DAL (p Frequency DAL (p Flow Framework (endent Modulators band-CSP (para_mu le Source Model GI trally Weighted CSI bowed Means (para Workspace] ned Movements via Disk] (lastapproach) Responses (lastap power/nolap (para bowe /holap (para bowe /holap (para)	Online Analysis (para_speccsp) andpower) (para_csp) nge (para_dal) ara_dal_hifreq) ara_dal_lofreq) para_dataflow) (para_modulators ultiband_csp) .M (para_multimod (para_speccsp) .M (para_speccsp) .Windowmeans) a CSP (lastapproach) a_bandpower) 3 (ver) 1 (lastapproach)	settings s) del) ch) lastapproa ach)	Help	(State	e of the	e art)
Cano	nical Motor Imag (la	istapproach)				-	_	

Using a More Ambitious Approach

A BCILAB 0.9			
Data Source Offline Analysis Online Analysis Settings Help			
Configure approach	;		
New sampling rate of the data	100		
Enoch time window relative to the events	[0.5.3.5]		
Event/marker types for which epochs shall be extracted	{'S 1' 'S 2'}		
Frequency-domain selection	[5 45]		
Number of CSP patterns (times two)	3		
Prior frequency weighting function	'@(t) t>=/ & t<=30'		
Machine learning function	lda 🗾		
Help	Cancel Ok		

Reviewing Results



Use Case B

- Question: Can we predict whether the user perceives an event as being an error?
- Experimental data: EEG, 32 channels, 2 sessions

Experimental Task

- Experimental task: ~100 randomized trials, 3 types of stimuli:
 - expected/correct event: type 'S 11'
 - unexpected event A: type 'S 12'
 - unexpected event B: type 'S 13'
- Sample:



Define approach

• This time, an ERP-specific approach is needed


Major customizations

📣 BCILAB 0	.9				
Data Source	Offline Analysis	Online Analysis	Settings	Help	

📣 Configure approach	
New sampling rate of the data	100
Epoch time window relative to the events Event/marker types for which epochs shall be extracted	[0 0.8] {'S 11' {'S 12' 'S 13'}}
Frequency-domain selection	[0.1 15]
Epoch intervals to take as features	4 0.45; 0.45 0.5; 0.5 0.55; 0.55 0.6]
Machine learning function	lda 🗾
Help	Cancel Ok

Train model, visualize



Train model, visualize



Using a Sparse Classifier



Using a Sparse Classifier

	📣 R	eview/edit approach			
BCILAB 0.9 Data Source Offlin	A	pproach properties ⊉↓ □ ● ‡ 문‡			
		ICA			A
		SurfaceLaplacian			
		FIRFilter			
		Projection			
		IIRFilter			
		Standardization			
		SparseReconstruction			
		 EpochExtraction 		\checkmark	
		TimeWindow	[0 0]	.8]	
		EventTypes	{'S :	1', {'S 12', 'S 13'}	}
		BaselineRemoval			
		WindowSelection			
		SpectralTransform			
		 SpectralSelection 		V	
		FrequencySpecification	[0.1	15]	
	E	Feature Extraction			
	E	FeatureExtraction			
		TimeWindows	[0.2	5 0.3;0.3 0.35;0.	35 0
	E	PluginFunctions	-		
		FeatureAdaptor	para	dataflow::@def	ault
		FeatureExtractor	para	windowmeans::	@ext
		FeatureAdaptorNeedsVoting		-	
		FeatureVisualizer	para	windowmeans::	@sho
		Machine Learning	P		
		Machinel earning			
			lda		
		Lambda	100		
		Rectinger	auto		
		WeightedBias	Guide		
Solacting lograginstand of Ida		WeightedCov			
Selecting logieg instead of lua				I	
for logistic regression) Learner Machine learning function. Applied to the data (features) produced within the paradigm; this is usually the last (and most adaptive) step in the processing from raw data to model or prediction.				paradigm; a to model	
		Help		Cancel	ОК

Using a Sparse Classifier

Selecting vb-ard Instead of vb for the "Variant" field

(yields "sparse logistic regression with variational Bayesian automatic relevance determination" as the classifier)

	Review/edit approach					
	Approach properties					
lin	≣ 2, ा वा व्य के क					
= 1	Darracceapracian	A				
	FIRFilter					
	Projection					
	IIRFilter					
	Standardization					
	SparseReconstruction					
	 EpochExtraction 					
	TimeWindow	[0 0.8]				
	EventTypes	{'S 11', {'S 12', 'S 13'}}				
	BaselineRemoval					
	WindowSelection					
	SpectralTransform					
	 SpectralSelection 	v				
	FrequencySpecification	[0.1 15]				
	Feature Extraction					
	FeatureExtraction					
	TimeWindows	[0.25 0.3;0.3 0.35;0.35 0				
	Pluga Functions					
Je	FeatureAdaptor	para_dataflow::@default				
	FeatureExcactor	para_windowmeans::@ext				
	FeatureAdaptoNeedsVoting					
	FeatureVisualizer	para_windowmeans::@sho				
	Machine Learning					
	MachineLearning					
	- Learner	logreg				
	Lambda	1				
	Variant	vb 🗸				
	Epsilon					
	Scaling	std				
	UseFallback					
	Variant Variant to use. Variational Bayes method unstable), or using a per-weight prior (ite Determination). Regularized methods: 11 sparse results	riant iant to use. Variational Bayes methods: using a joint uniformative prior (possibly itable), or using a per-weight prior (iter), or using a sparse prior (Automatic Relevance ermination). Regularized methods: 11 for sparse results, or 12 for not necessarily irse results				
	Help	Cancel OK				

Training, Visualizing



Training, Visualizing

 Sparse classifiers can give more robust models (fewer channels / sources of errors used), and more interpretable models (only the most relevant features retained)

Next: Basic Scripting Tour



• Applying a Spec-CSP approach as seen in the GUI:

```
% load the data set
traindata = io_loadset('data:/tutorial/imag_movements1/calib/DanielS001R01.dat');
% define the approach
myapproach = {'SpecCSP' 'SignalProcessing', {'EpochExtraction', ...
{'TimeWindow', [0 3.5], 'EventTypes', {'StimulusCode_2', 'StimulusCode_3'}}};
% learn a predictive model
[trainloss,lastmodel,laststats] = bci train({'data',traindata,'approach',myapproach}); %#ok<>
```

```
% visualize results
bci_visualize(lastmodel)
```



Visualization Output





• Running the resulting model in real time on some data:

```
% load feedback session
testdata = io_loadset('data:/tutorial/imag_movements1/feedback/DanielS001R01.dat');
```

```
% play it back in real time
run_readdataset('Dataset',testdata);
```

```
% process data in real time using lastmodel, and visualize outputs run_writevisualization('Model',lastmodel, 'VisFunction','bar(y)');
```

Script Output

BCILAB 0.9





 Doing a parameter search and nested crossvalidation





• Running the advanced ERP analysis (with sparse classifier):

```
% define markers; here, two groups of markers are being defined; the first group represents class 1
% (correct responses), and the second group represents class 2 (incorrect responses).
mrks = {{'S101', 'S102'}, {'S201', 'S202'}};
% define ERP windows of interest; here, 7 consecutive windows of 50ms length each are being
% specified, starting from 250ms after the subject response
wnds = [0.25 \ 0.3; 0.3 \ 0.35; 0.35 \ 0.4; 0.4 \ 0.45; 0.45 \ 0.5; 0.5 \ 0.55; 0.55 \ 0.6];
% define load training data (BrainVision format)
traindata = io_loadset('data:/tutorial/flanker_task/12-08-001_ERN.vhdr');
% define approach
mvapproach = {'Windowmeans' ...
     SignalProcessing', {'EpochExtraction', {'TimeWindow', [0 0.8], 'EventTypes', mrks}, 'SpectralSelection', [0.1 15]}, ...
    'FeatureExtraction',{'TimeWindows',wnds}, ...
    'MachineLearning',{'Learner', {'logreg', [],'Variant','vb-ard'}}};
%learn model
[trainloss,lastmodel,laststats] = bci_train({'data',traindata,'approach',myapproach});
% visualize results
bci_visualize(lastmodel)
```



Visualization Output





• Running a batch analysis for 3 modern approaches and 136 data sets (upcoming version only):

% define markers; here, two groups of markers are being defined; the first group represents class 1
% (correct responses), and the second group represents class 2 (incorrect responses).
mrks = {{'S101','S102'},{'S201','S202'}};
wnds = [0.25 0.3;0.3 0.35;0.35 0.4; 0.4 0.45;0.45 0.5;0.5 0.55;0.55 0.6];

% define approaches

```
approaches.wmeans_lda = {'Windowmeans' 'flt',{'events',mrks,'epoch',[0 0.8],'spectrum',[0.1 15]},'fex',{'wnds',wnds}};
approaches.wavelet_lars = {'Dataflow' 'flt',{'events',mrks,'epoch',[0 0.8],'spectrum',[0.1 15],'wavelet','on'},...
'ml',{'learner',{'logreg',[],'variant','lars'}};
approaches.dal = {'DAL_Lofreq','SignalProcessing',{'Resampling',60,'IIRFilter','off','FIRFilter',[0.1 0.5 18 21], ...
'EpochExtraction',{'EventTypes',mrks,'TimeWindow',[-0.2 0.65]}},'MachineLearning',{'Learner',{'dal',2.^(8:-0.125:1)}}};
```

% run a batch analysis... results = bci_batchtrain('Datasets','/data/projects/grainne/ERN/*.vhdr','Approaches',approaches,'RetainExistingResults',true);

Sample Plugins



BCILAB Architecture





FFT Filter

```
function signal = flt_fft(varargin)
% Apply an FFT to each epoch of an epoched signal (Example).
% Signal = flt_fft(Signal, LogPower)
%
% This is example code to transform a signal into the power domain, or log-power domain. A
% fully-featured version of this is flt_fourier.
%
% In:
%
              Epoched data set to be processed
    Signal :
%
%
    LogPower : whether to take the logarithm of the power (instead of the raw power) (default: false)
%
% Out:
%
    Signal : processed data set
%
%
                                 Christian Kothe, Swartz Center for Computational Neuroscience, UCSD
%
                                 2011-01-19
if ~exp_beginfun('filter') return; end
% requires epoched data, works best on spatially filtered data
declare_properties('name','EpochedFFT', 'depends','set_makepos', 'follows',{'flt_project','flt_window'}, 'independent_channels',true);
% declare arguments
arg_define(varargin,...
    arg_norep({'signal','Signal'}), ...
    arg({'do_logpower', 'LogPower'}, false, [], 'Compute log-power. Taking the logarithm of the power in each frequency band is easier to
% apply FFT and cut mirror half of the resulting samples
tmp = fft(signal.data,[],2);
tmp = tmp(:,1:signal.pnts/2,:);
% take signal power or log(power)
if do_logpower
    signal.data = log(abs(tmp));
else
    signal.data = abs(tmp);
end
exp_endfun;
```



Kernel SVMs (via SVMperf)

arg_define([O 3],varargin, ...

arg_norep('trials'), ...

arg_norep('targets'), ...

arg({'cost', 'Cost'}, search(2.^(-5:2:15)), [], 'Regularization parameter. Reasonable range: 2.^(-5:2:15), greater is stronger. By default, it is average arg({'ptype', 'Type'}, 'classification', {'classification', 'regression', 'ranking'}, 'Type of problem to solve.', 'cat', 'Core Parameters'), ...

arg({'kernel', 'Kernel'}, 'rbf', {'linear', 'rbf', 'poly', 'sigmoid', 'user'}, 'Kernel type. Linear, or Non-linear kernel types: Radial Basis Functions (gene arg({'g', 'RBFScale', 'gamma'}, search(2.^(-16:2:4)), [], 'Scaling parameter of the RBF kernel. Should match the size of structures in the data; A reasonal arg({'d', 'PolyDegree'}, uint32(3), [], 'Degree for the polynomial kernel.', 'cat', 'Core Parameters'), ...

arg({'etube', 'EpsilonTube', 'tube'}, 0.1, [], 'Epsilon tube width for regression.', 'cat', 'Core Parameters'), ...

arg({'rbalance','CostBalance','balance'}, 1, [], 'Relative cost of per-class errors. The factor by which training errors on positive examples outweight

arg({'s','SigmoidPolyScale'}, 1, [], 'Scale of sigmoid/polynomial kernel.','cat','Miscellaneous'), ...

arg({'r', 'SigmoidPolyBias'}, 1, [], 'Bias of sigmoid/polynomial kernel.', 'cat', 'Miscellaneous'), ...

arg({'u','UserParameter'}, '1', [], 'User-defined kernel parameter.','cat','Miscellaneous','type','char','shape','row'),

arg({'bias','Bias'}, false, [], 'Include a bias term. Only implemented for linear kernel.','cat','Miscellaneous'), ...

arg({'scaling','Scaling'}, 'std', {'none','center','std','minmax','whiten'}, 'Pre-scaling of the data. For the regulariation to work best, the features : arg({'clean','CleanUp'}, false, [], 'Remove inconsistent training examples.','cat','Miscellaneous'), ...

arg({'epsi','Epsilon','eps'}, 0.1, [], 'Tolerated solution accuracy.','cat','Miscellaneous'), ...

arg({'verbose', 'Verbose'}, false, [], 'Show diagnostic output.', 'cat', 'Miscellaneous'));

if is_search(cost)

cost = 1; end
if is_search(q)

q = 0.3; end

% find the class labels

classes = unique(targets); if length(classes) > 2 % in this case we use the voter model = ml_trainvote(trials,targets,'1v1',@ml_trainsvmlight,@ml_predictsvmlight,varargin{:}); else % scale the data

scale the data
sc_info = hlp_findscaling(trials,scaling);
trials = hlp_applyscaling(trials,sc_info);

% remap target labels to -1,+1

```
targets(targets==classes(1)) = -1;
targets(targets==classes(2)) = +1;
```

```
% rewrite sme string args to numbers
```

ptype = hlp_rewrite(ptype,'classification','c','regression','r','ranking','p'); %#ok<*NODEF>
kernel = hlp_rewrite(kernel,'linear',0,'poly',1,'rbf',2,'sigmoid',3,'user',4);

% build the arguments

args = sprintf('-z %s -c %f -v %d -w %f -j %f, -b %d -i %d -e %f -t %d -d %d -g %f -s %f -r %f -u %s', ... ptype,cost,verbose,etube,rbalance,bias,clean,epsi,kernel,d,g,s,r,u);

% run the command

```
model = svmlearn(trials,targets,args);
model.sc_info = sc_info;
model.classes = classes;
```



classdef ParadigmCSP < ParadigmDataflowSimplified</pre>

CSP Paradigm

```
methods
        function defaults = preprocessing defaults(self)
            defaults = {'FIRFilter', {'Frequencies', [6 8 28 32], 'Type', 'minimum-phase'}, 'EpochExtraction', [0.5 3.5], 'Resampling', 10'
        end
        function [model,needsvoting] = feature adapt(self,varargin)
            arg define (varargin, ...
                arg norep('signal'), ...
                arg({'patterns', 'PatternPairs'},3,[], 'Number of CSP patterns' (times two).', 'cat', 'Feature Extraction', 'type', 'expres:
            if signal.nbchan < patterns</pre>
                error('CSP requires at least as many channels as you request output patterns. Please reduce the number of pattern pa:
            for k=1:2
                trials{k} = exp eval(set picktrials(signal, 'rank', k), 1);
                covar{k} = cov(reshape(trials{k}.data,size(trials{k}.data,1),[])');
                covar{k}(~isfinite(covar{k})) = 0;
            end
            [V,D] = eig(covar{1}, covar{1}+covar{2});
            model.filters = V(:,[1:patterns end-patterns+1:end]);
            P = inv(V);
            model.patterns = P([1:patterns end-patterns+1:end],:);
            model.chanlocs = signal.chanlocs;
            needsvoting = true;
        end
        function features = feature extract(self, signal, featuremodel)
            features = zeros(size(signal.data,3),size(featuremodel.filters,2));
            for t=1:size(signal.data,3)
                features(t,:) = log(var(signal.data(:,:,t)'*featuremodel.filters)); end
        end
        function layout = dialog layout defaults(self)
            layout = {'flt.srate','flt.epoch','flt.fir.fspec','flt.fir.ftype',[],'pred.fad.patterns',[],'pred.ml.learner'};
        end
    end
end
```



Ongoing Work

- Better domain-specific assumptions in BCI approaches (moving beyond off-the-shelf components)
 - e.g., expressed as general convex optimization, Bayesian inference
- Integration of (quantitative) prior knowledge
 - Anatomical (or even functional) priors from brain atlases, etc.
- Integration of larger data sources
 - Multiple recordings, multiple subjects, ...
- Better exploitation of multiple modalities
 - Hybrid BCIs, general cognitive state assessment, ...
- More hardware devices! 🙂



Teaser: Some Model Types

 Window1 (0.25s to 0.3s)
 Window2 (0.3s to 0.35s)
 Window3 (0.35s to 0.4s)

 Window4 (0.4s to 0.45s)
 Window5 (0.45s to 0.5s)
 Window6 (0.5s to 0.55s)

 Window7 (0.55s to 0.6s)
 Window5 (0.45s to 0.5s)
 Window6 (0.5s to 0.5s)

VB-ARD (on ERPs)











Thanks! Questions?

http://sccn.ucsd.edu/wiki/BCILAB