Swartz
Center for
Computational
Neuroscience

Lecture 6: BCILAB Toolbox Anatomy

Introduction to Modern Brain-Computer Interface Design

Christian A. Kothe
SCCN, UCSD

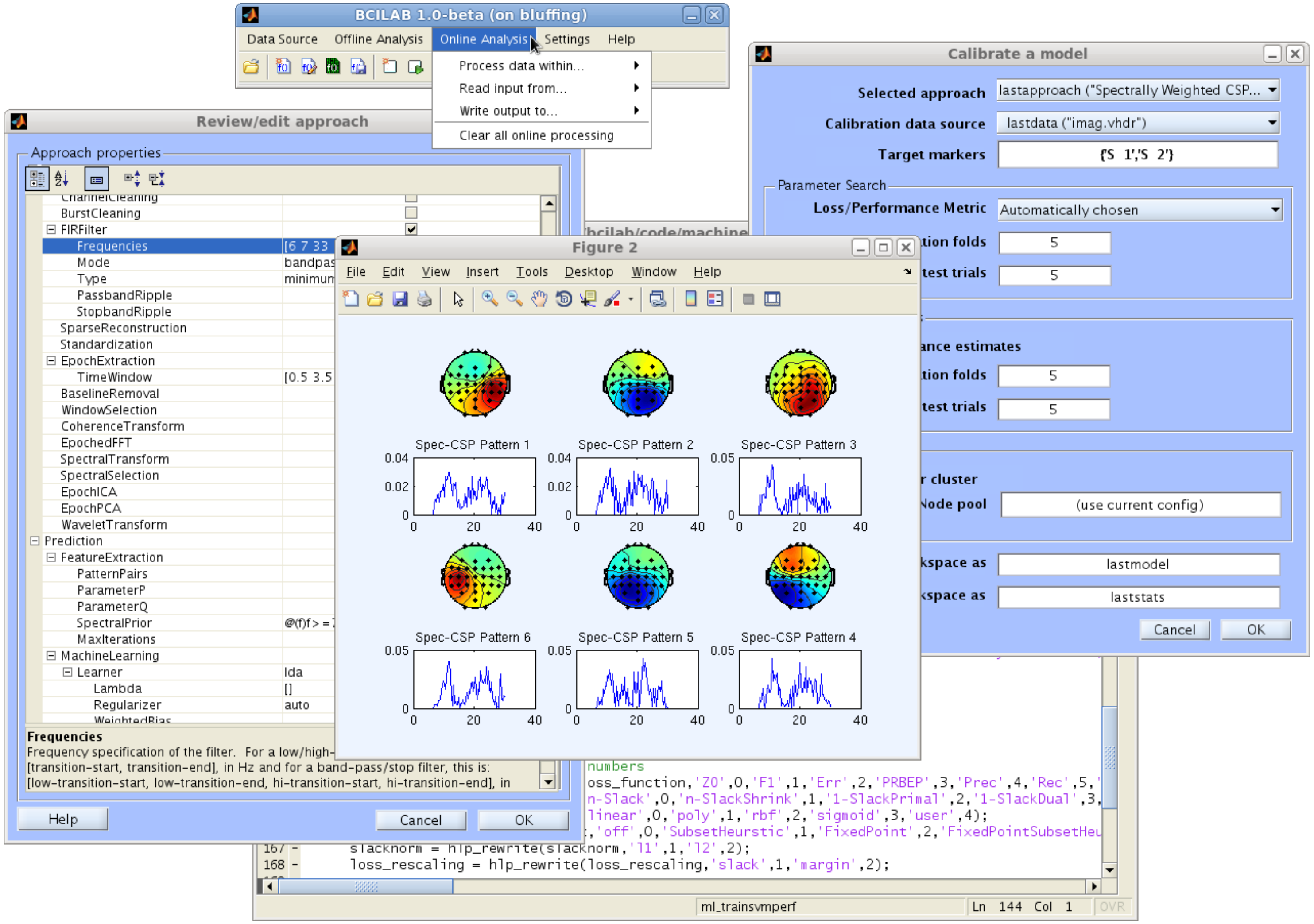
Outline

1. Context and Background
2. Quick Methods Teaser
3. Architecture Overview
4. Plugin Concepts
5. Data Representations and Pipeline

6.1 Context and Background

Swartz
Center for
Computational
Neuroscience

The BCILAB Toolbox



Approach properties:

```
  ChannelCleaning
  BurstCleaning               □
⊟ FIRFilter                   ☑
    Frequencies        [6 7 33
    Mode               bandpas
    Type               minimum
    PassbandRipple
    StopbandRipple
  SparseReconstruction
  Standardization
⊟ EpochExtraction
    TimeWindow         [0.5 3.5
  BaselineRemoval
  WindowSelection
  CoherenceTransform
  EpochedFFT
  SpectralTransform
  SpectralSelection
  EpochICA
  EpochPCA
  WaveletTransform
⊟ Prediction
  ⊟ FeatureExtraction
      PatternPairs
      ParameterP
      ParameterQ
      SpectralPrior    @(f)f>=
      MaxIterations
  ⊟ MachineLearning
    ⊟ Learner          lda
        Lambda         []
        Regularizer    auto
        WeightedBias
```

Frequencies
Frequency specification of the filter. For a low/high-
[transition-start, transition-end], in Hz and for a band-pass/stop filter, this is:
[low-transition-start, low-transition-end, hi-transition-start, hi-transition-end], in

Calibrate a model:

Selected approach	lastapproach ("Spectrally Weighted CSP...
Calibration data source	lastdata ("imag.vhdr")
Target markers	{'S 1','S 2'}
Loss/Performance Metric	Automatically chosen
tion folds	5
test trials	5
ance estimates	
tion folds	5
test trials	5
r cluster	
Node pool	(use current config)
kspace as	lastmodel
kspace as	laststats

```
numbers
oss_function,'ZO',0,'F1',1,'Err',2,'PRBEP',3,'Prec',4,'Rec',5,'
n-Slack',0,'n-SlackShrink',1,'1-SlackPrimal',2,'1-SlackDual',3,
linear',0,'poly',1,'rbf',2,'sigmoid',3,'user',4);
;,'off',0,'SubsetHeurstic',1,'FixedPoint',2,'FixedPointSubsetHeu
slacknorm = hlp_rewrite(slacknorm,'l1',1,'l2',2);
loss_rescaling = hlp_rewrite(loss_rescaling,'slack',1,'margin',2);
```

ml_trainsvmperf   Ln 144 Col 1   OVR

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

ftp://sccn.ucsd.edu/pub/bcilab

Context

- 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, …)

Software Environment For:

- **Brain-Computer Interface Design** (Cognitive Monitoring)
- **Methods Research**:
  - Design & rapid prototyping of new methods & methods from literature
  - Offline testing, performance evaluation & batch comparison
  - Simulated online testing
- **Rapid Prototyping**:
  - Real-time use
  - Prototype deployment

Facts & Figures

- Developed since 2010 at SCCN, UCSD (primarily by me)
- Precursor was the PhyPA toolbox (Kothe & Zander, 2006-’09)
- Built on top of EEGLAB (Delorme & Makeig, 2004)
- The Largest open-source BCI toolbox by methods and algorithms (2012)
- Offline and online processing both in MATLAB, same code base, cross-platform, 32/64bit

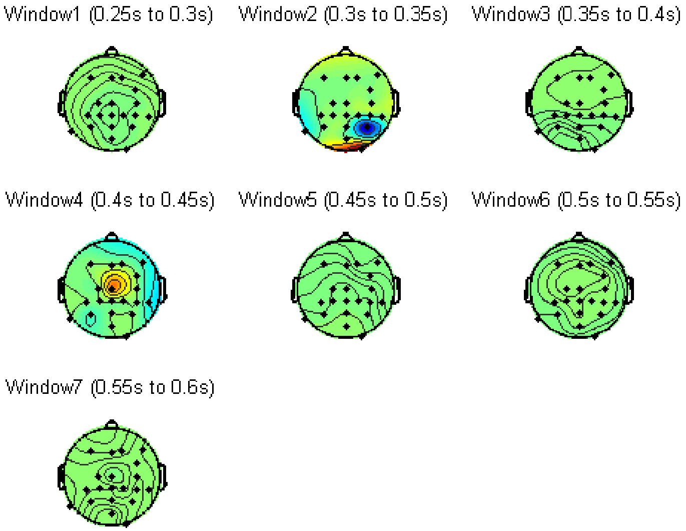
Further Goals

- Provide large array of *existing methods* to reproduce existing literature – e.g., in benchmarking and comparison studies
- Provide *state-of-the-art and novel methods* to rapidly set up well-performing BCIs
- Provide plugin frameworks and backend solvers to implement new methods quickly
- GUI for beginners & experimenters, scripting for experts and MATLAB veterans – largely the same feature set
- Allow for both conventional designs (e.g., data flow) and for radically new approaches

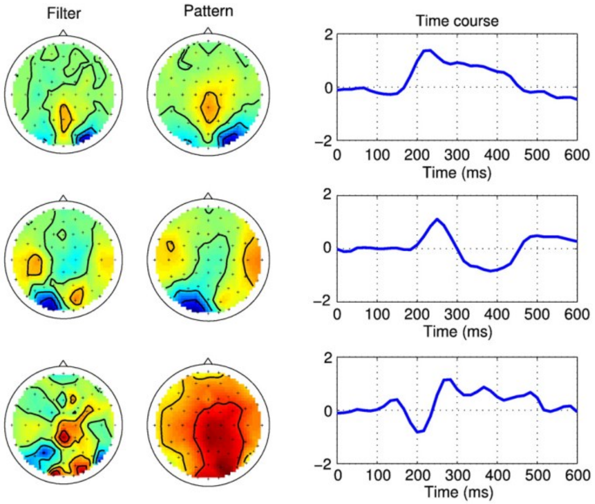
6.2 Quick Methods Teaser

Time-Domain / ERP Baseline

Windowed Means



DAL-ERP



(*)

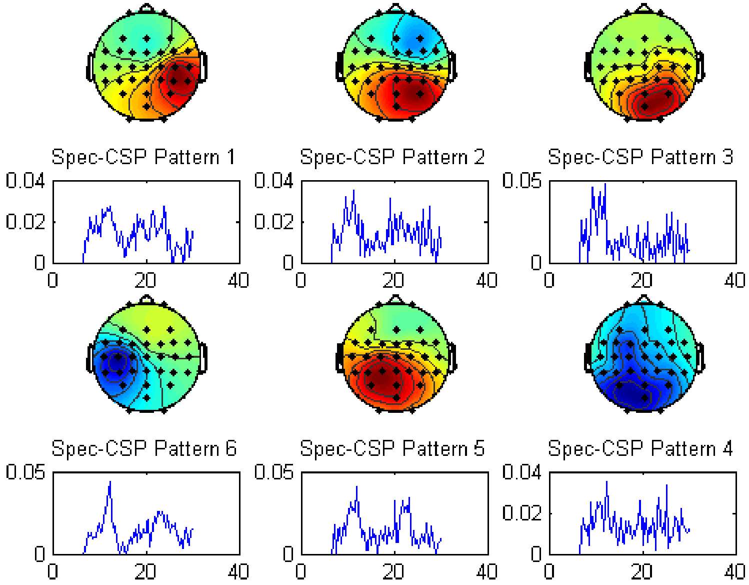
- Traditional linear classifier for event-locked brain responses, usually using LDA
- Time windows manually assigned
- Examples: error recognition, surprise

- State-of-the-art approach, no hand-tuned parameters
- Uses rank-regularized logistic or linear regression

(*image: Tomioka et al., 2010)

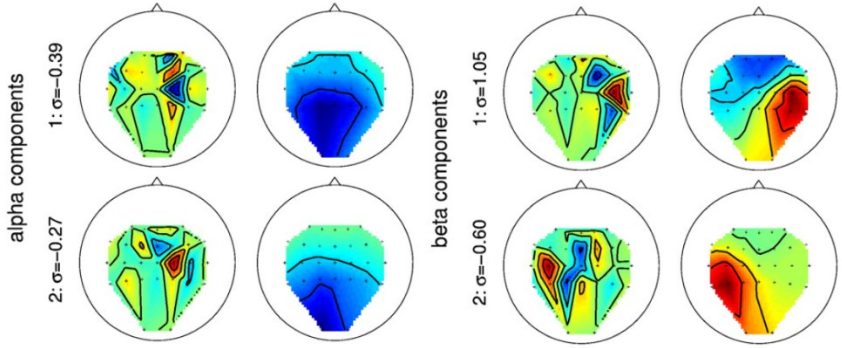
Oscillatory Processes Baseline

Common Spatial Patterns Family



- Filter-Bank CSP (FBCSP): multiple bands/windows
- Diagonal Loading CSP (DLCSP): cov. shrinkage
- Composite CSP (CCSP): covariance prior
- Tikhonov-regularized CSP (TRCSP): filter shrinkage
- Spectrally weighted CSP (Spec-CSP): learning spectral filters from the data

DAL-OSC



(*)

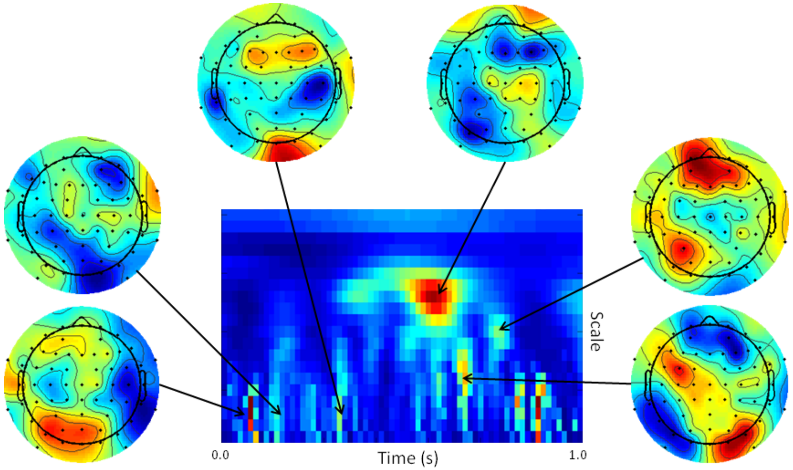
- State-of-the-art approach, no hand-tuned parameters
- Also uses rank-regularized logistic or linear regression
- Single-step approach, jointly optimal

(*image: Tomioka et al., 2010)

New Methods

Methods for Time-Domain Analysis

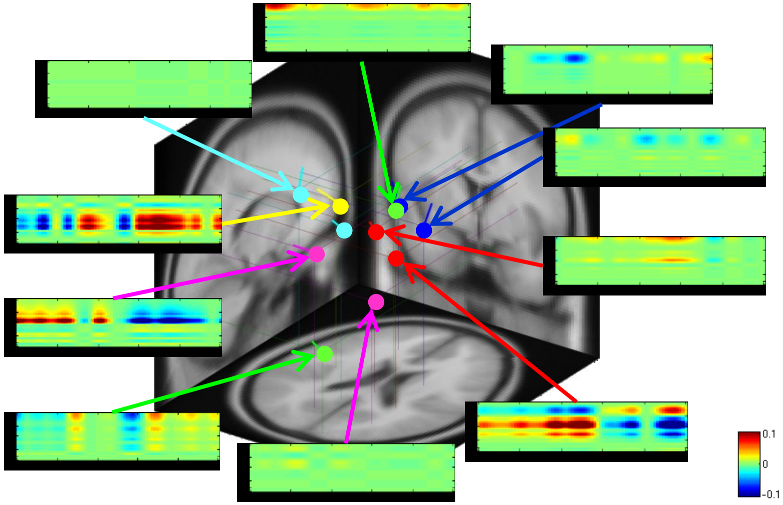
(below: Wave Propagation Imaging)



- Classify event-locked brain responses
- Best methods to date learn optimal evolving spatial filters (as above)
- Several methods in the same performance ballpark
- Examples: error recognition, surprise
- Benchmark paper in preparation

Methods for Oscillatory Analysis

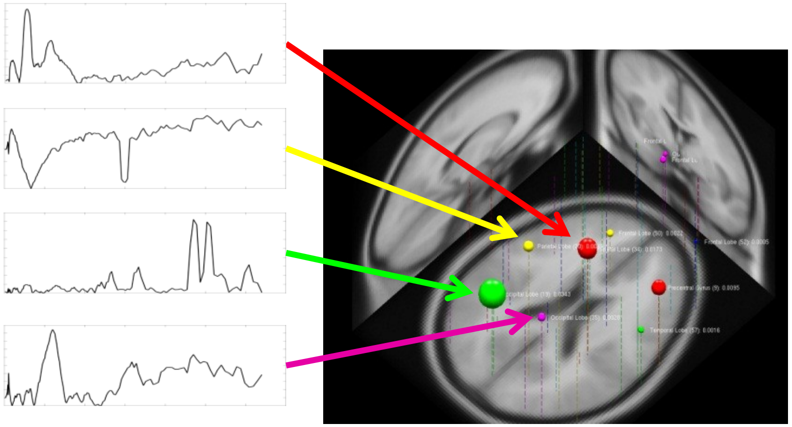
(below: Regularized Spatio-Spectral Dynamics)



- Applicable to slowly-changing operator state and background activity as well as event-related transients
- RSSD is a pioneering method for learning full source-level time/frequency structure
- Examples: cognitive load, attention shifts
- Presented at ICON’11; methods and data papers in preparation

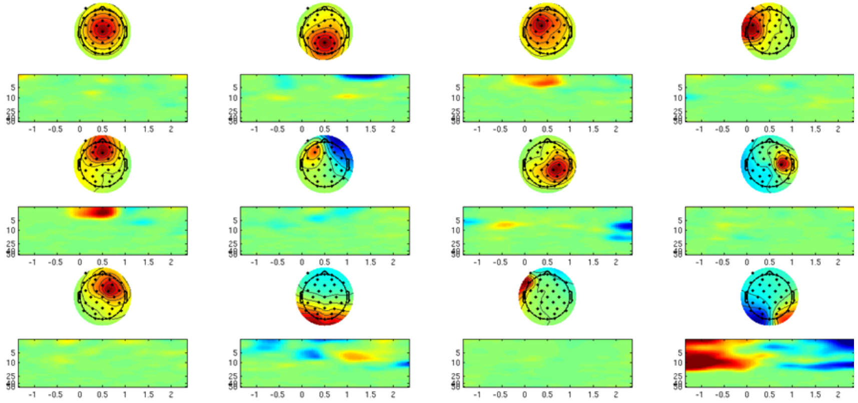
New Methods (Exploratory)

Overcomplete Spectral Regression



- Long-term stationary oscillations
- Can integrate information from a corpus of data (across persons)
- Examples: fatigue, alertness, sleep stages
- Presented at EMBC'11
- Related method presented at ABCI'11

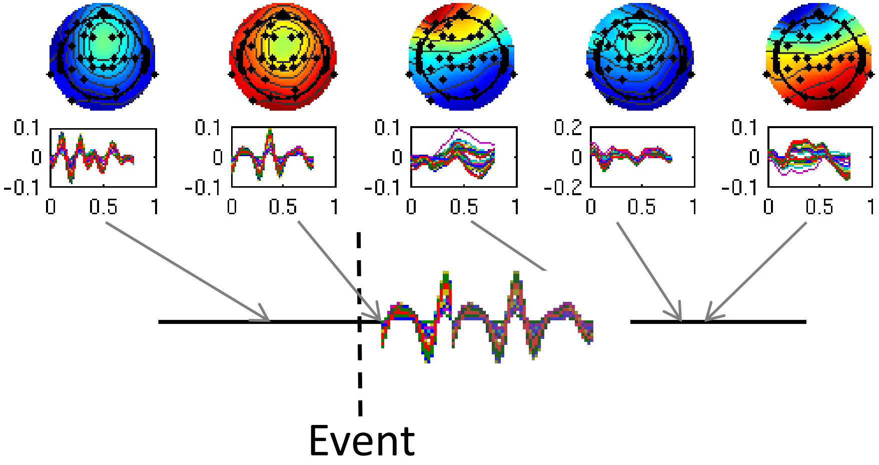
Spatio-Spectral Bayes



- A fully Bayesian version of RSSD aimed at neuroscientific modeling
- Allows for extensive statistical analysis of results
- Presented at Sloan-Swartz '11

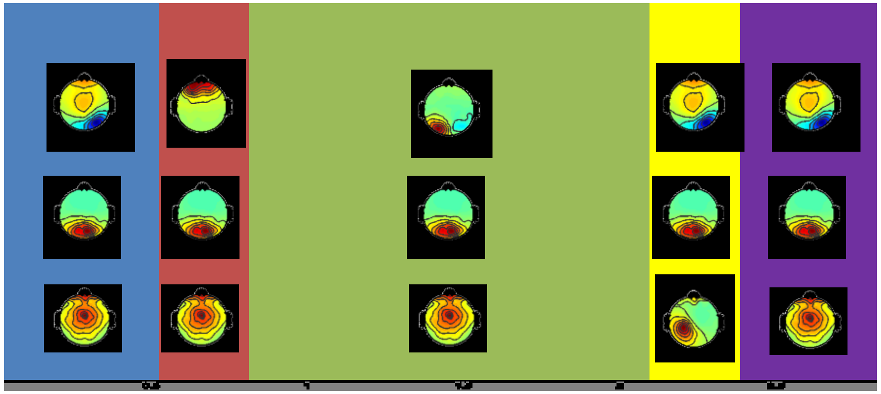
New Methods (Exploratory!!)

Pattern Alignment Learning



- Finds time-jittered brain processes associated with known events in the work environment
- Radically new approach using joint optimization
- Applications: target event detection and other event-related cognitive responses

Independent Component Sparse Decomposition & Others

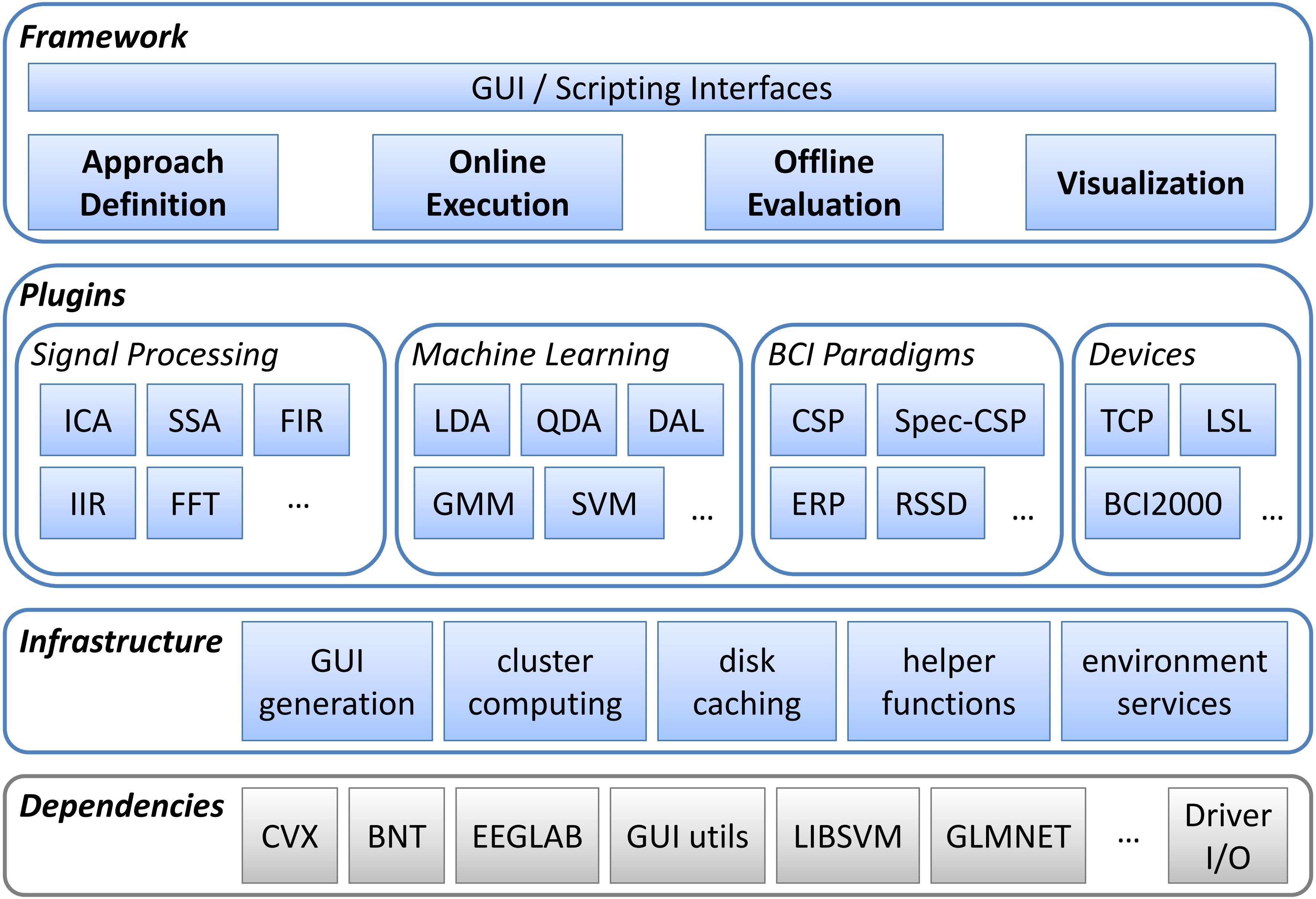


- General-purpose method for finding cortical source constellations that produce ongoing scalp signals
- Lifts restrictions on the number of source processes under consideration
- Using old code, needs to be updated
- Other methods: auto-regressive modeling of joint human-system interface dynamics

(*image: Bigdely-Shamlo)

6.3 Architecture Overview

Toolbox Layers



Framework

GUI / Scripting Interfaces

Approach Definition

Online Execution

Offline Evaluation

Visualization

Plugins

Signal Processing

ICA SSA FIR IIR FFT …

Machine Learning

LDA QDA DAL GMM SVM …

BCI Paradigms

CSP Spec-CSP ERP RSSD …

Devices

TCP LSL BCI2000 …

Infrastructure

GUI generation

cluster computing

disk caching

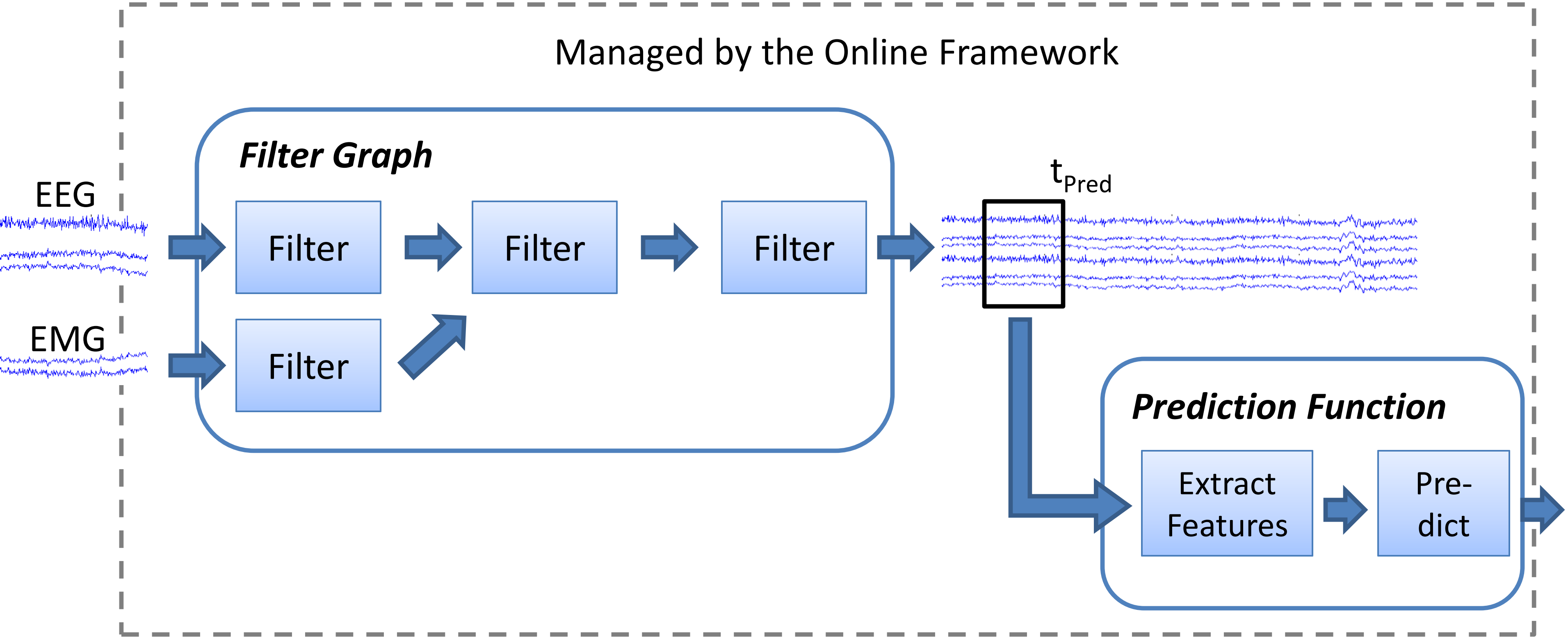
helper functions

environment services

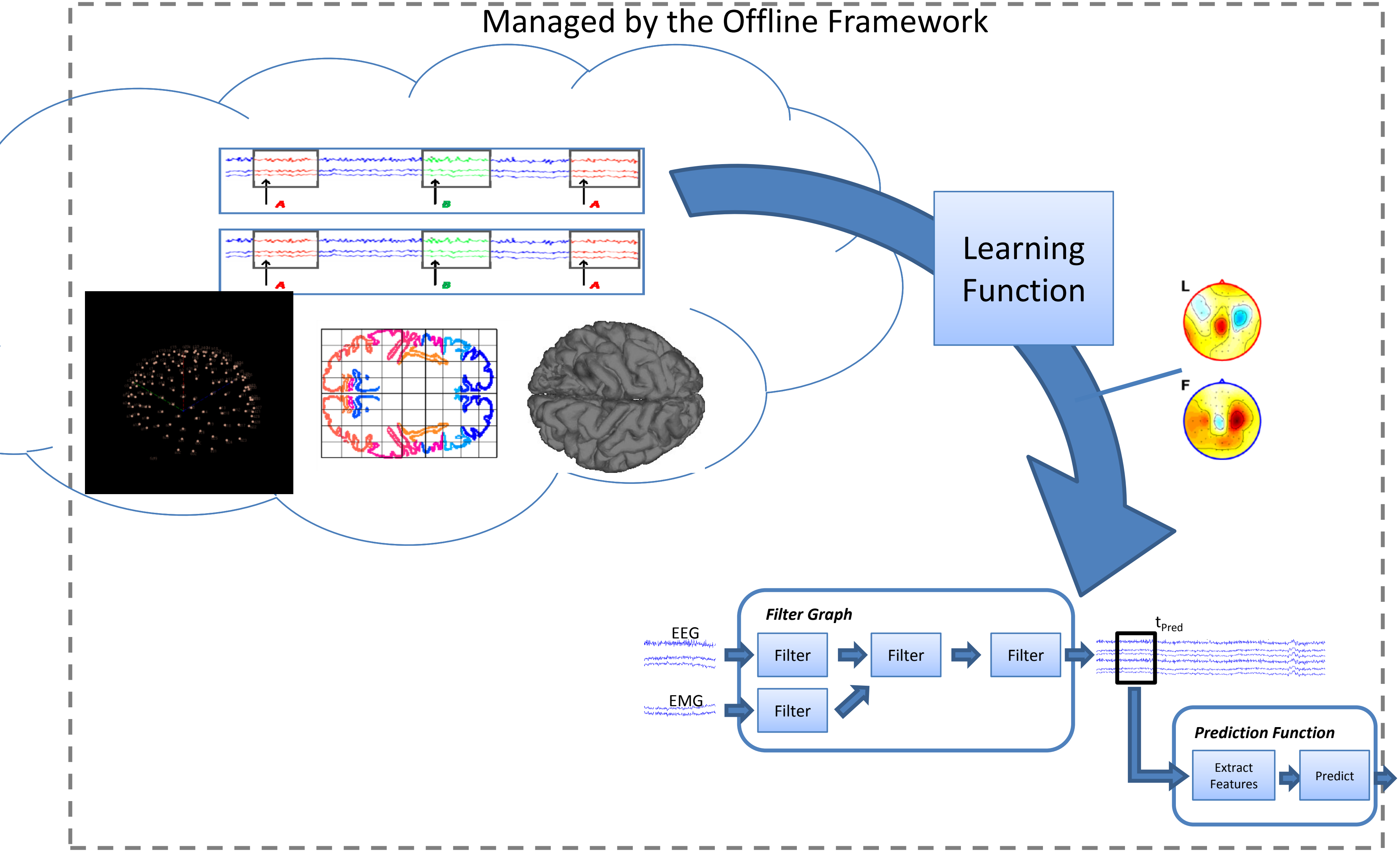
Dependencies

CVX BNT EEGLAB GUI utils LIBSVM GLMNET … Driver I/O

Scope of the Online Framework



Scope of the Offline Framework



Managed by the Offline Framework

Learning Function

Filter Graph

Filter

Filter

Filter

Filter

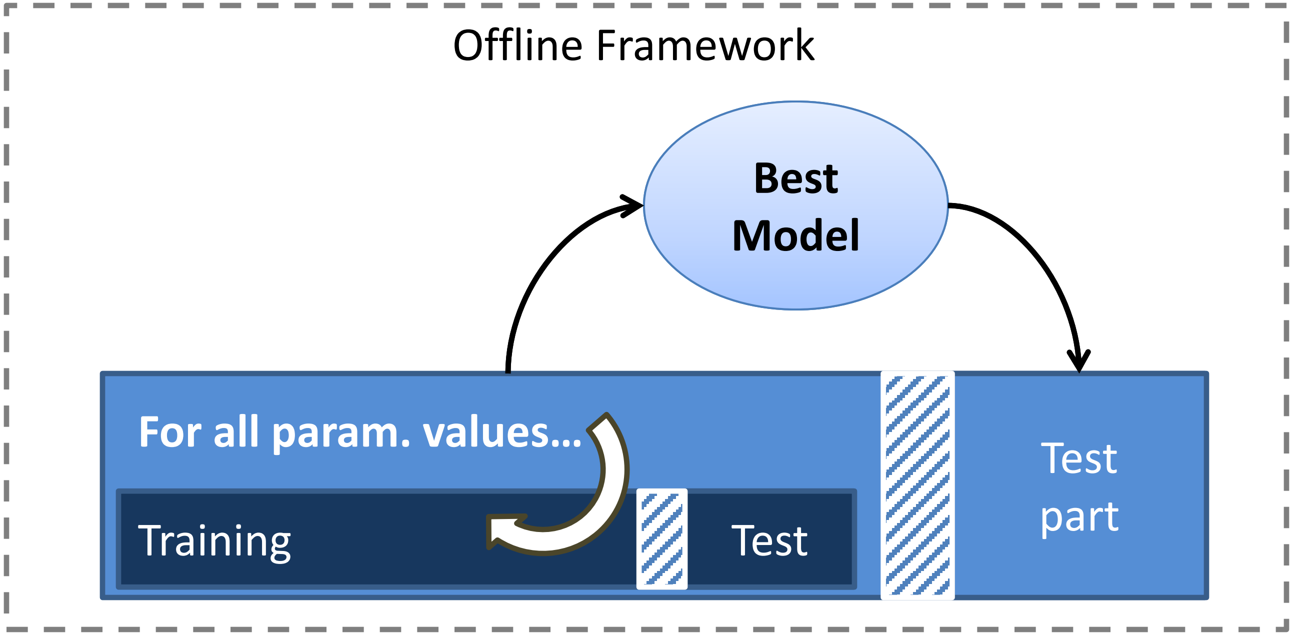
Prediction Function

Extract Features

Predict

Scope of the Offline Framework

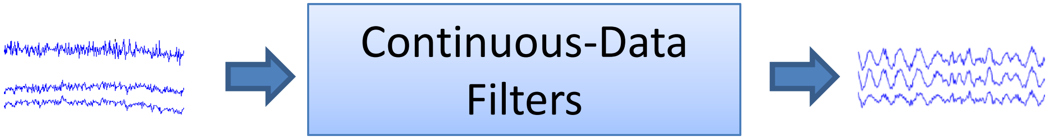
- **Also Covered:** Cross-validation, Grid Search, Nested Cross-Validation



6.4 Plugin Concepts

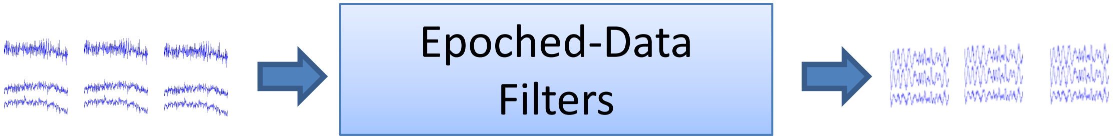
Plugin Concepts: Filters

- Filters can operate on continuous signals…



Continuous-Data Filters

- … or on segmented (“epoched”) signals:



Epoched-Data Filters

Plugin Concepts: Filters

- *Static (“stateless”) filters*:

```
EEG = flt_selchans(EEG,{‘C3’,’C4’,’Cz’})
```

- *Dynamic (“stateful”) filters*:

```
[EEG,State] = flt_resample(EEG,200,State)
```

- *Epoched filters*:

```
EEG = flt_fourier(EEG)
```

Plugin Concepts: Filters

- **Evil caveat:** filters have *lazy evaluation behavior*, i.e. they do not evaluate unless forced:

```
EEG = flt_fourier(EEG)
>> EEG =
      head: @flt_fourier
     parts: {[1x1 struct]}
  codehash: '356d73563c38107c63a33762cc7789ba'
```

Not what you wanted!

Plugin Concepts: Filters

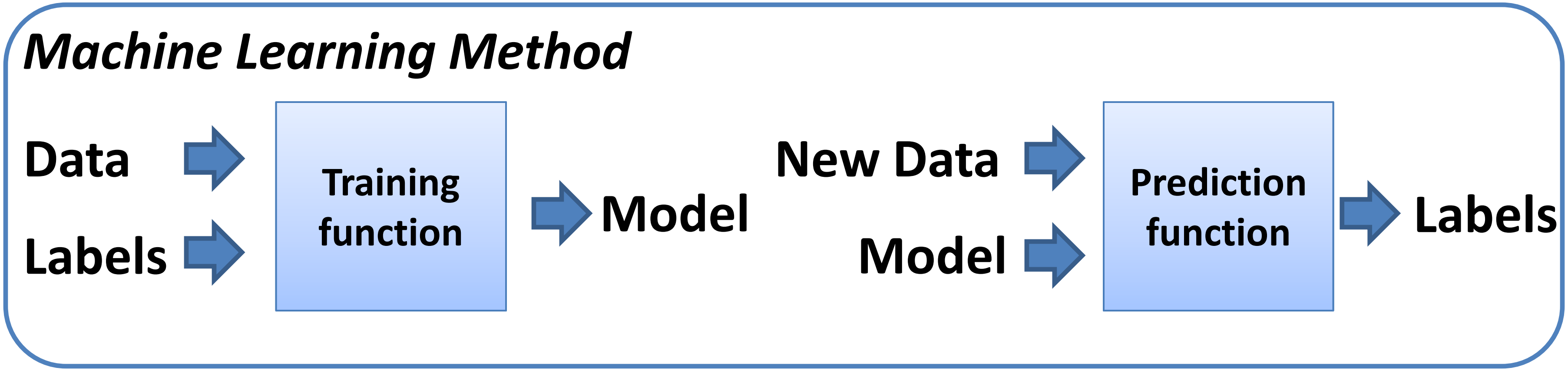
- **Evil caveat:** filters have lazy evaluation behavior, i.e. they do not evaluate unless forced:

```
EEG = exp_eval(flt_fourier(EEG))
```

The right way

Plugin Concept: Machine Learning

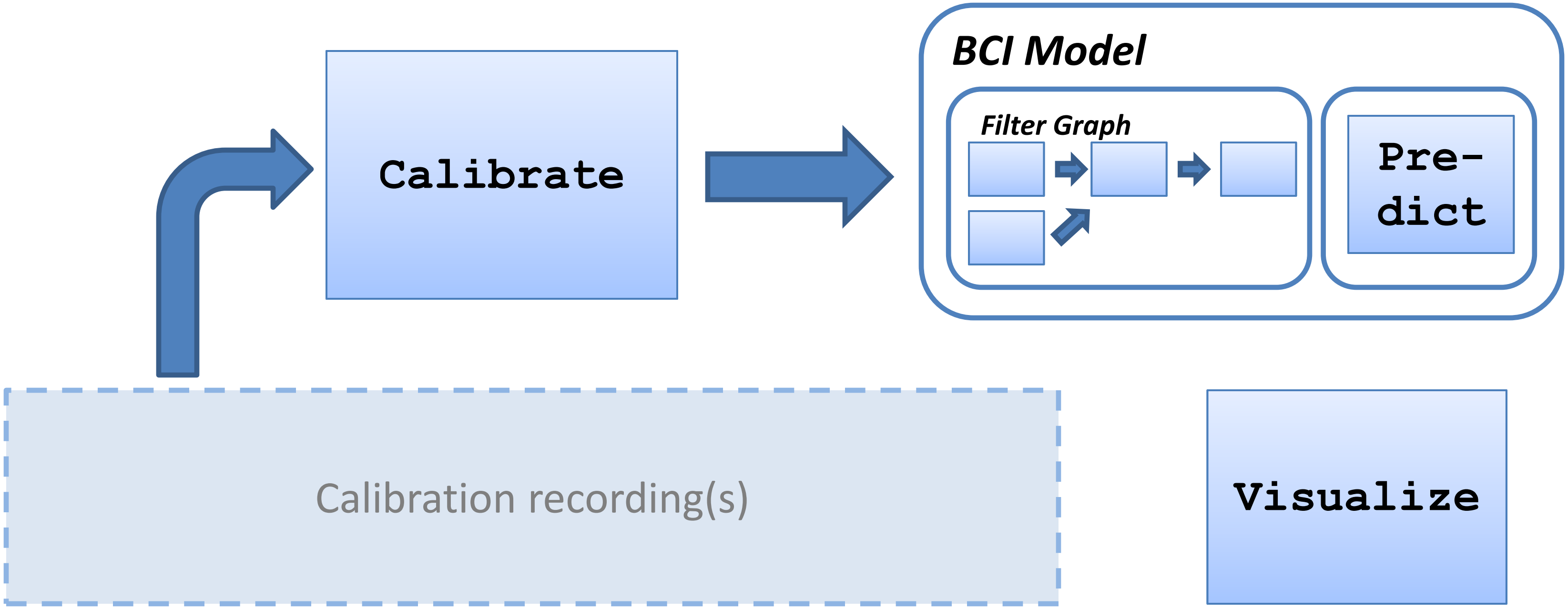
- Machine learning functions come in pairs:



```
M = ml_trainlda(X,y)
p = ml_predictlda(Xnew,M)
```

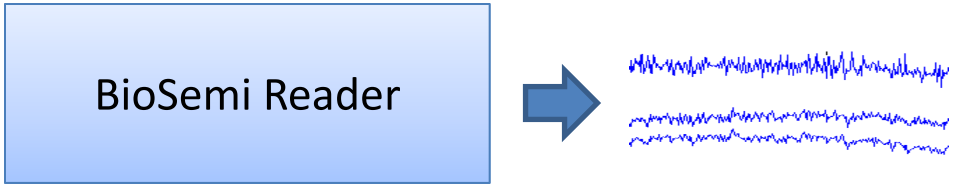
Plugin Concepts: Paradigms

- **BCI paradigms** are the coarsest plugin type in BCILAB and *tie all parts of a BCI approach together* (signal processing, feature extraction, machine learning, …)
- They are invoked by the offline/online framework



Plugin Concepts: Online Readers

- Online reader plugins read signals from a source device and make them available in the MATLAB workspace:



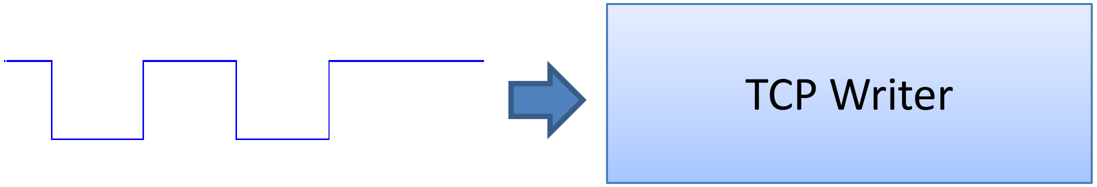
BioSemi Reader

- Example:

```
run_readbiosemi();
```

Plugin Concepts: Online Writers

- Online writer plugins write BCI outputs (i.e., predictions) to some external destination:

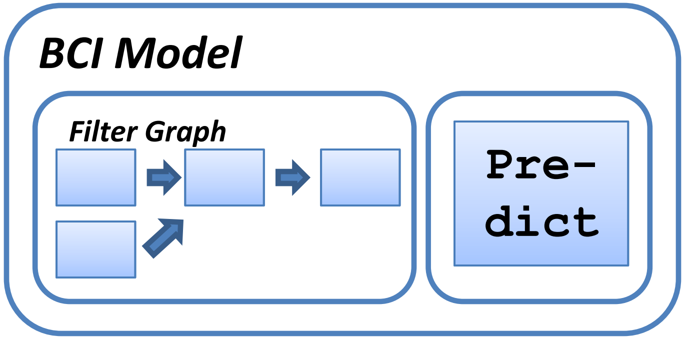


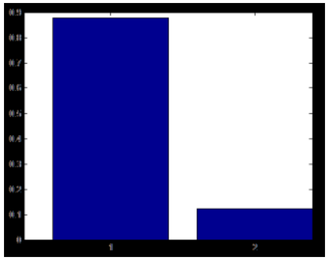
- Example:

```
run_writetcp(‘mdl',‘strm','192.168.1.5',12467)
```

6.5 Data Representations and Pipeline

Data Representations

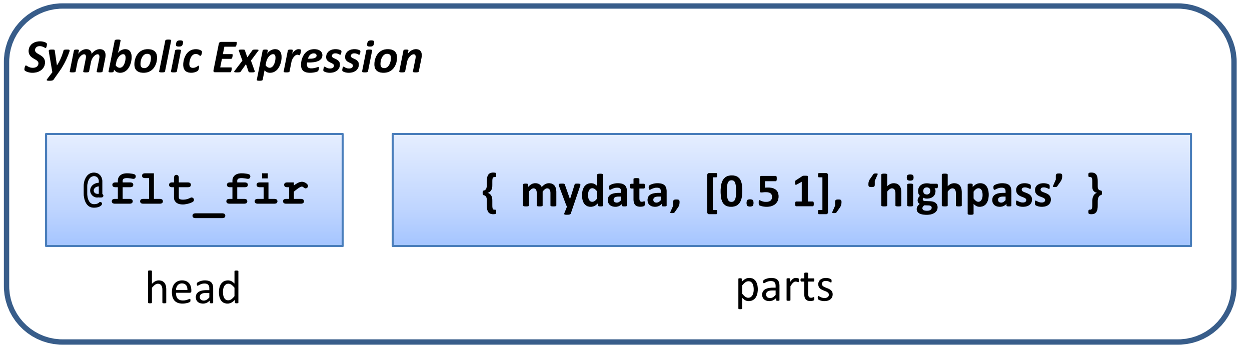




Probability Distributions

$$\begin{bmatrix} f_1 \\ f_2 \\ \vdots \end{bmatrix}$$

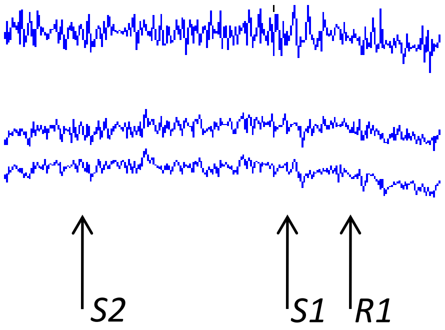
Feature Vectors



Data Representations

Signal

.data

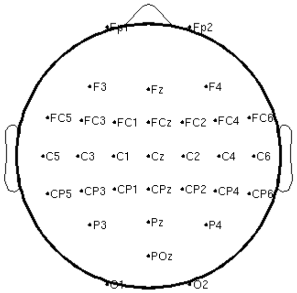


.event

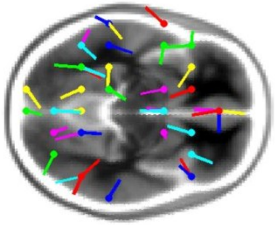
.srate *200Hz*

.xmin *0.0s*

.chanlocs



.dipfit



... (meta-data)

Signal Bundle

.streams

- Signal 1
- Signal 2
- ⋮
- Signal n

... (meta-data)

Dataset Collection

- Bundle 1
- Bundle 2
- ⋮
- Bundle n

Pipeline Notion

- BCILAB is a framework that resembles a processing pipeline: first configure everything, then apply it to one or more data sets
- **Configuration Inputs:**
  - Mapping between marker type strings and numeric class labels
  - Base BCI Paradigm to execute – “what to run?”
  - Custom parameters for the paradigm
  - Evaluation Scheme – “how to run it?” (e.g., what type of cross-validation)

Swartz Center for Computational Neuroscience

Pipeline Processes

- **Curate:** bring the input data into standard form
- **Design:** define the computational approach
- **Train:** invoke all steps necessary for training (calibrating) a BCI and estimates performance
- **Predict:** apply a BCI to some data offline
- **Visualize:** visualize BCI model internals
- **Run Online:** apply a BCI online / incrementally
- **Batch Analysis:** perform a series of processing steps, optionally in parallel

Training Algorithm

1. Train optimized model on entire data
   - Optionally with parameter search

2. Optional: do a cross-validation on entire data to quantify the model performance
   - Optionally with nested parameter search

A Note on Data Curation

- Up-front conversion of data set and file format idiosyncrasies into uniform representation:
  - Continuous data – unfiltered, possibly re-referenced
  - Correct channel labels/locations
  - Correct event types, latencies, etc
  - Other common meta-data about raw recordings
- Usually done in a first pass before any BCILAB function is touched

L6 Questions?