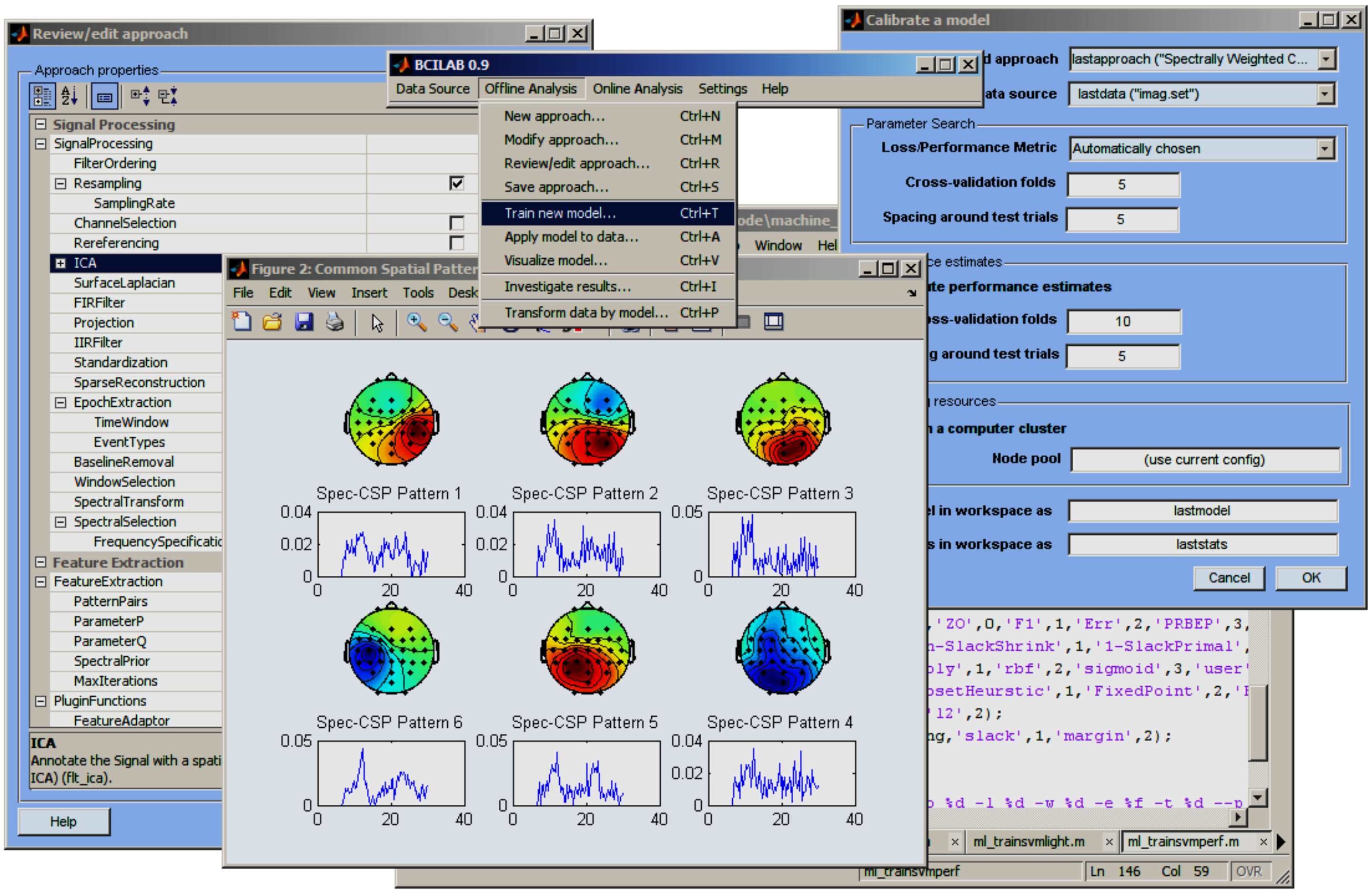
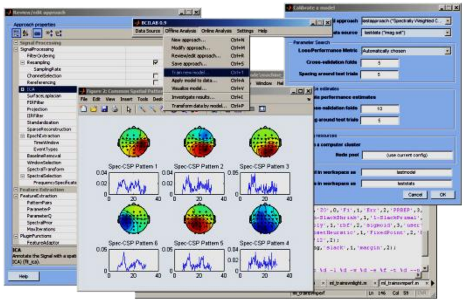
Swartz Center for Computational Neuroscience

BCILAB



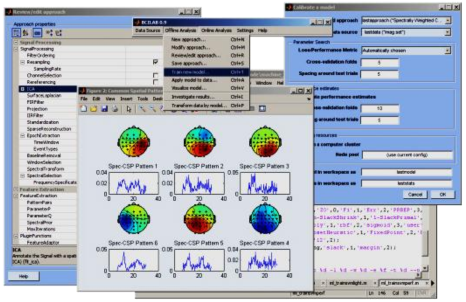
[http://sccn.ucsd.edu/**wiki/BCILAB**](http://sccn.ucsd.edu/wiki/BCILAB)



Summary

- Software environment for:
  - Design & rapid prototyping of cognitive monitoring systems
  - Offline testing, performance assessment
  - Simulated online testing
  - Real-time use, prototype deployment

Features



- Largest collection of machine learning and signal processing methods for BCI / CSA publicly available (open source)
- Currently ca. 130 algorithms:
  - conventional BCI components (CSP, LDA, logBP, Spec-CSP, SVMs, …)
  - state-of-the-art approaches (DSLR, AMICA, HKL, DPGMM, …)
  - new approaches (RSSD, OSR, ICSD, SSB, WPI, …)
  - fast/general backend solvers (DAL, glm-ie, CVX, …)
- All fully integrated

BCILAB Components

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
- OSC
- BCI2000
- ...

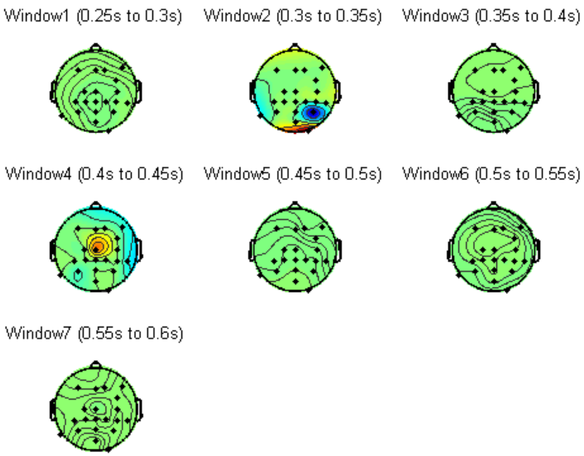
Infrastructure

- GUI generation
- cluster computing
- disk caching
- helper functions
- environment services

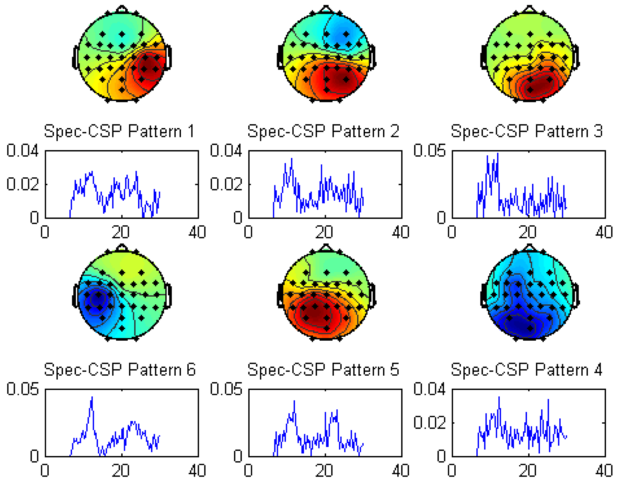
Dependencies

- CVX
- BNT
- EEGLAB
- GUI utils
- LIBSVM
- GLMNET
- ...
- Driver I/O

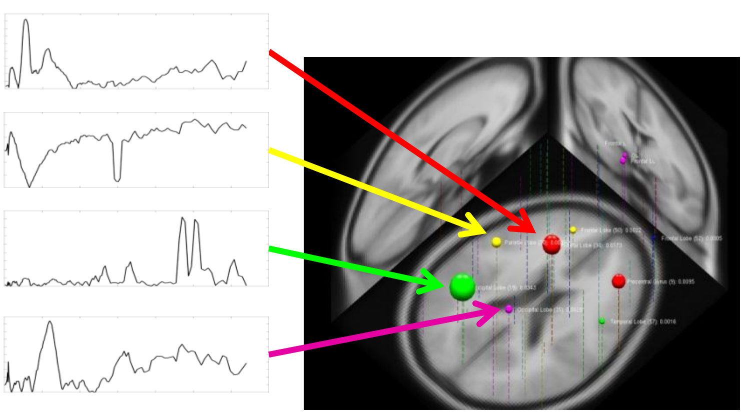
Some Model Types Visualized



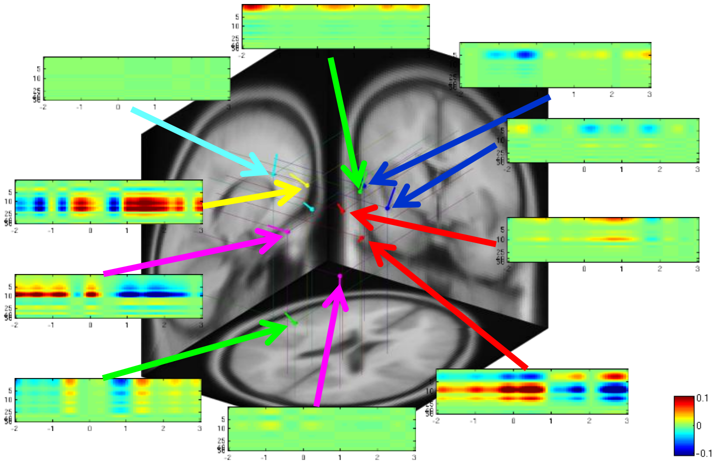
VB-ARD (on ERPs)



Spec-CSP



OSR



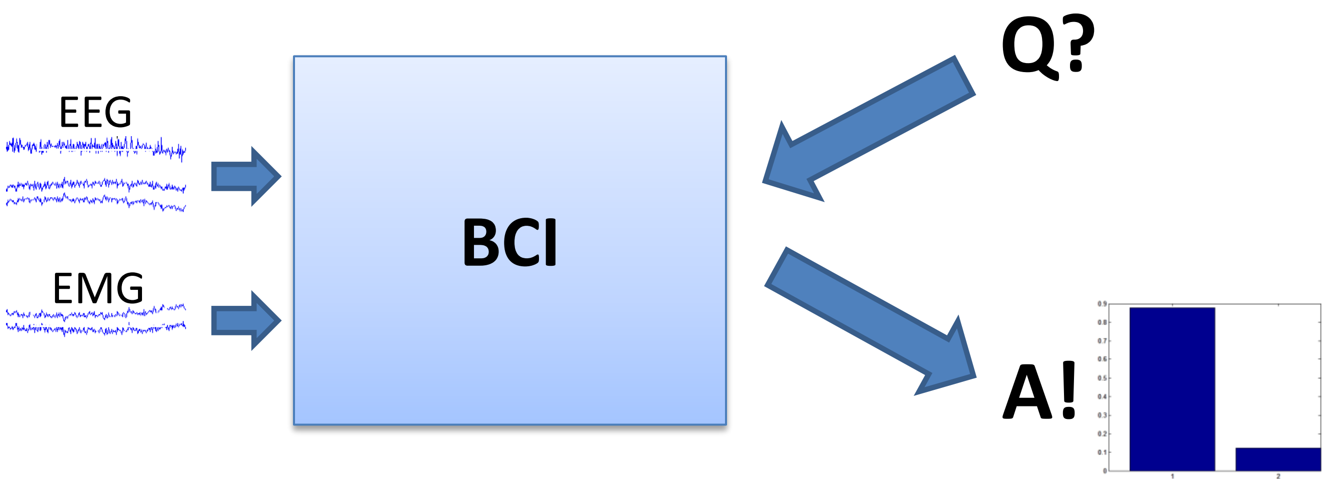
RSSD

Framework

- Fully automated pipeline (artifact rejection, caching, filtering, parallelization, parameter search, cross-validation, cloud deployment)
- Fully probabilistic framework
- Neuroscience-aware features (anatomical constraints, source-level analysis, …)
- Support for batch processing
- Support for corpus-scale analysis (multiple sessions, persons, etc.)

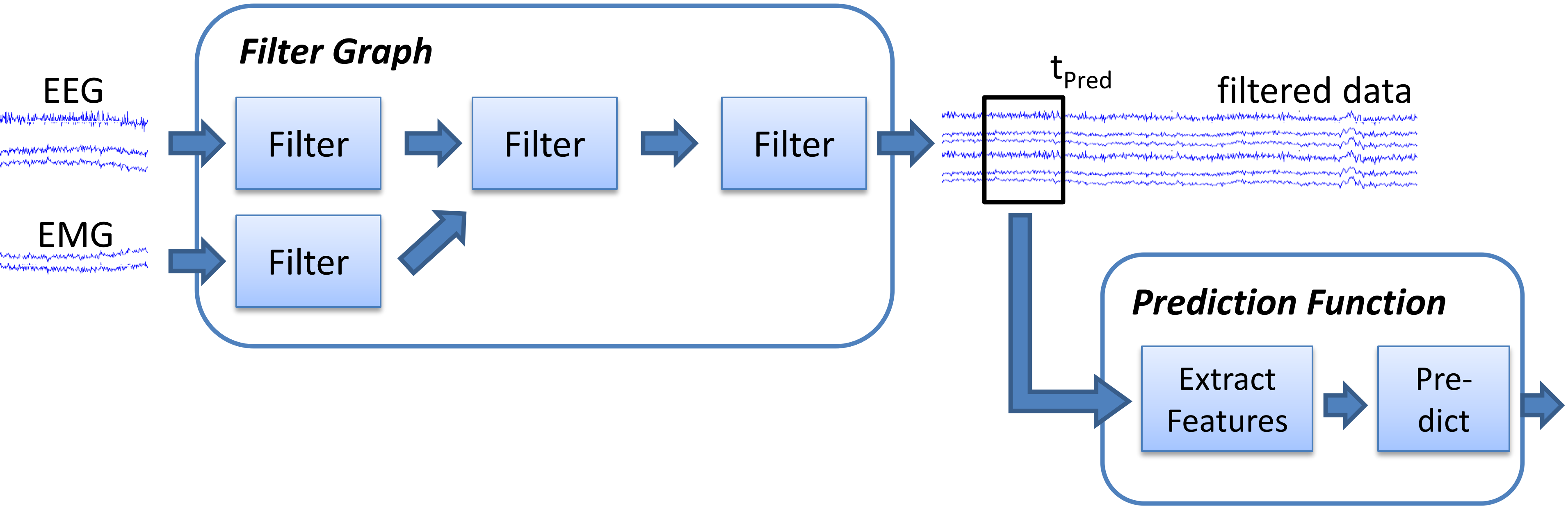
BCI Metaphor

- BCIs in BCILAB are acting as an oracle that consumes one or more biosignals and can respond to (pre-defined) queries about cognitive state



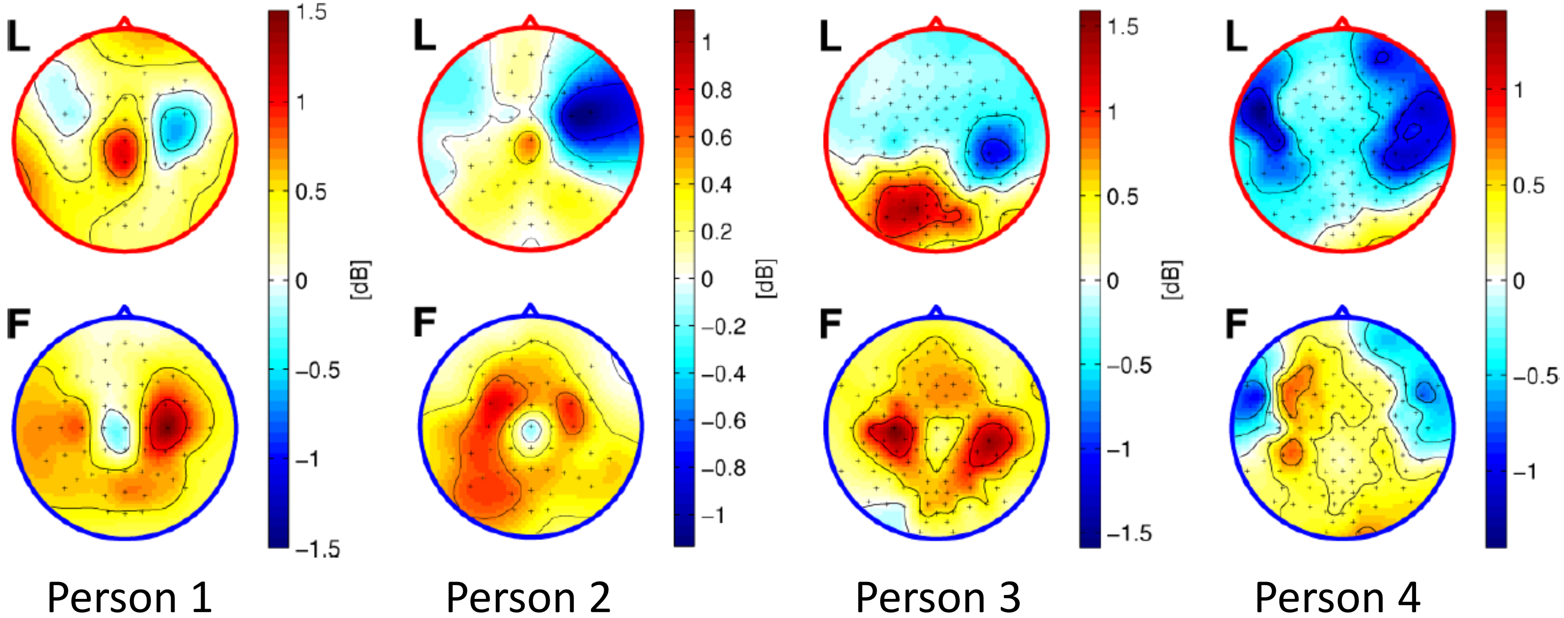
Online Data Flow

- A filter graph receives all input samples and produces pre-filtered data
- The prediction function may be queried on demand on the filter graph's outputs



Model Calibration

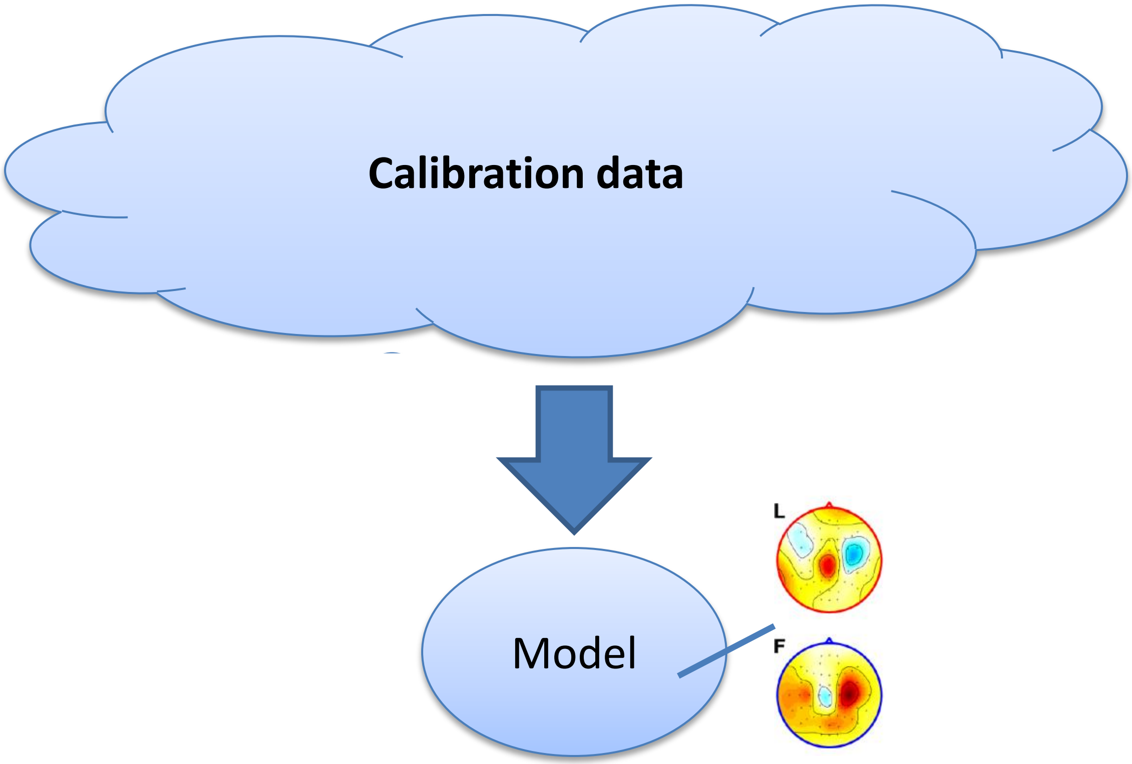
- **Problem**: optimal parameters for a BCI model depend on person, montage, task, etc.



(image: Blankertz et al. 2007)

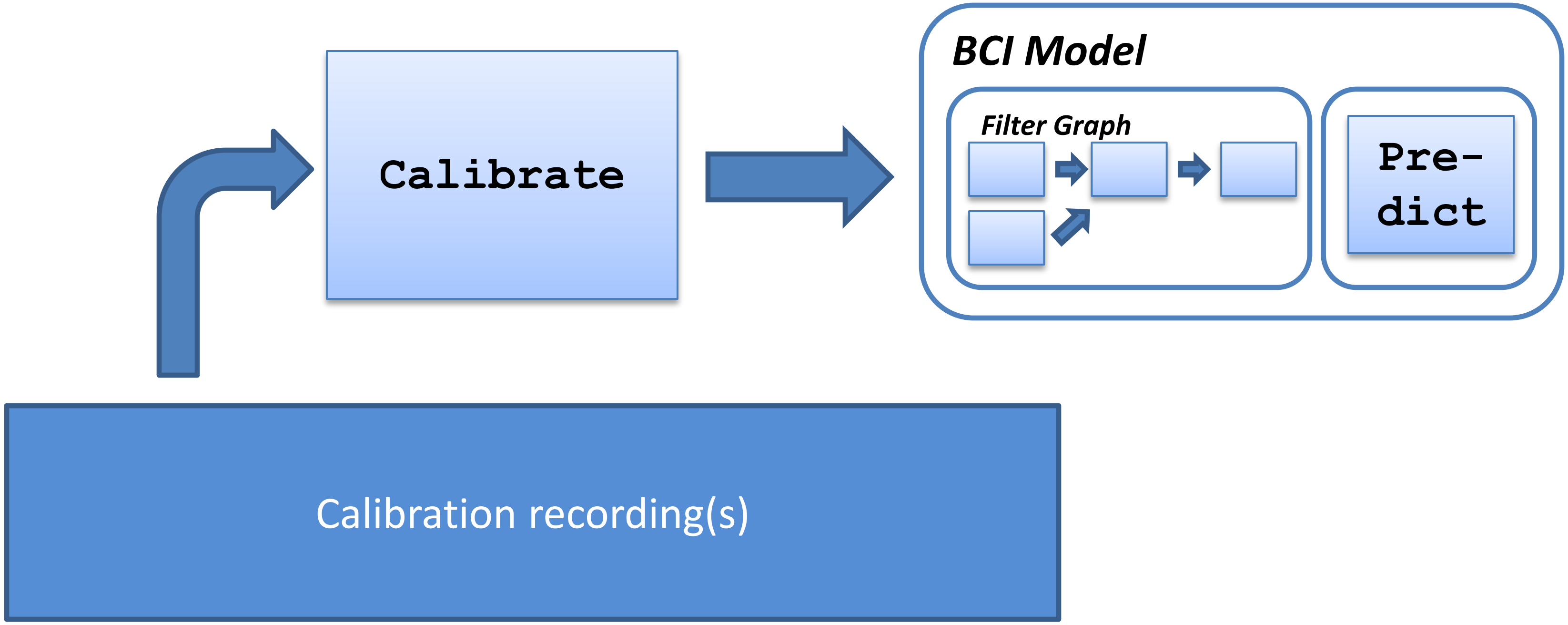
Model Calibration

- Therefore infer model parameters from calibration / training data



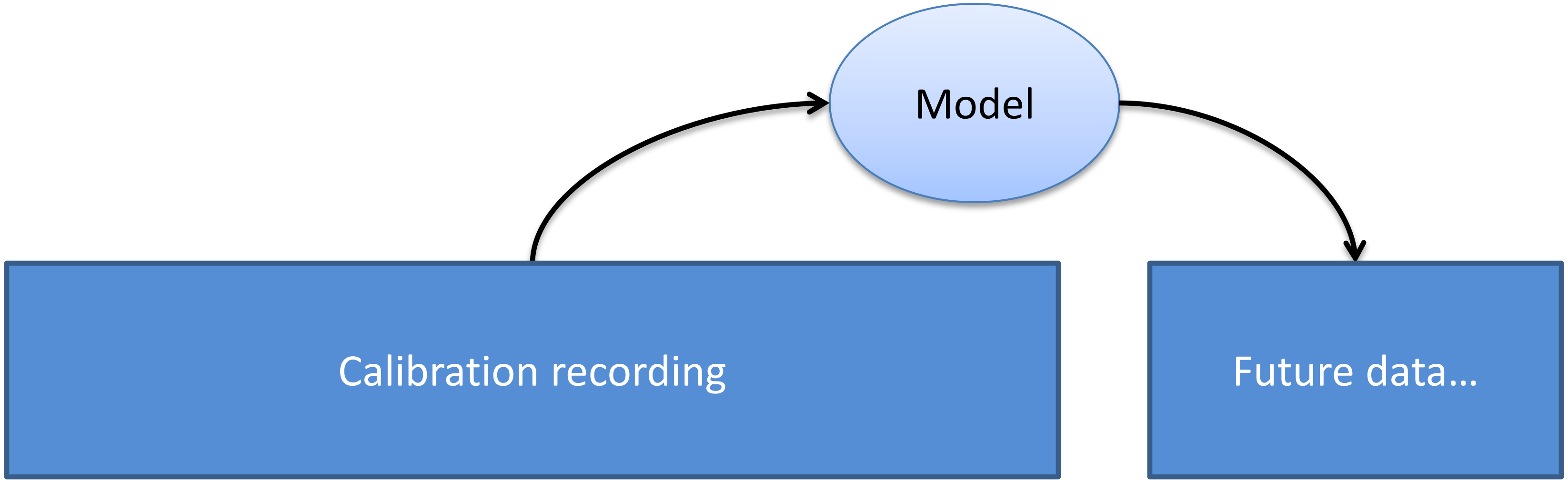
Calibration: BCI Paradigms

- BCI paradigms are the coarsest plugin type in BCILAB and tie all parts of a BCI approach together
- They are seeds for new BCI designs and cornerstones of BCILAB usage



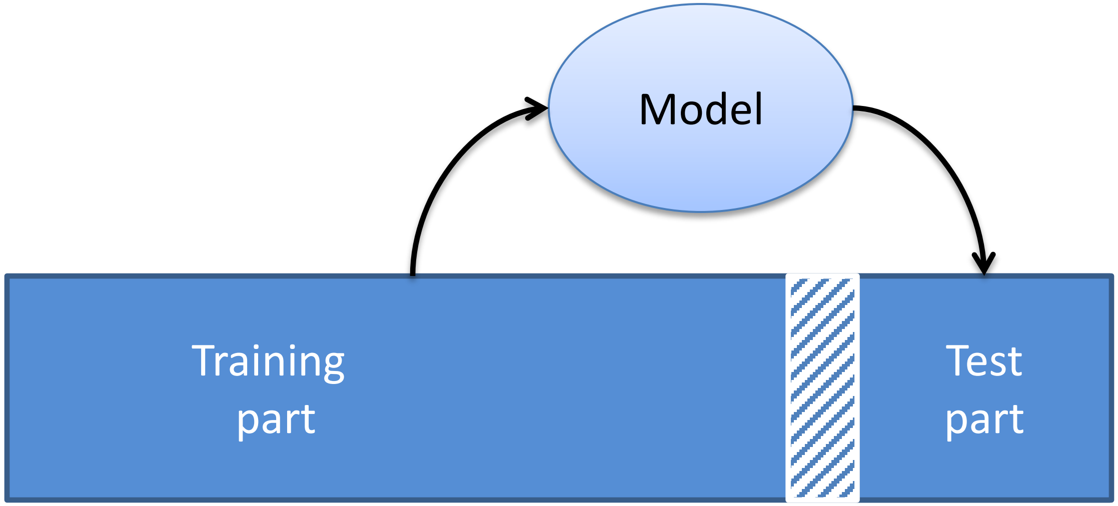
Evaluation: Offline

- Given calibration data
- Estimate model parameters (spatial filters, statistics)
- Apply the model to new data (online / single-trial)



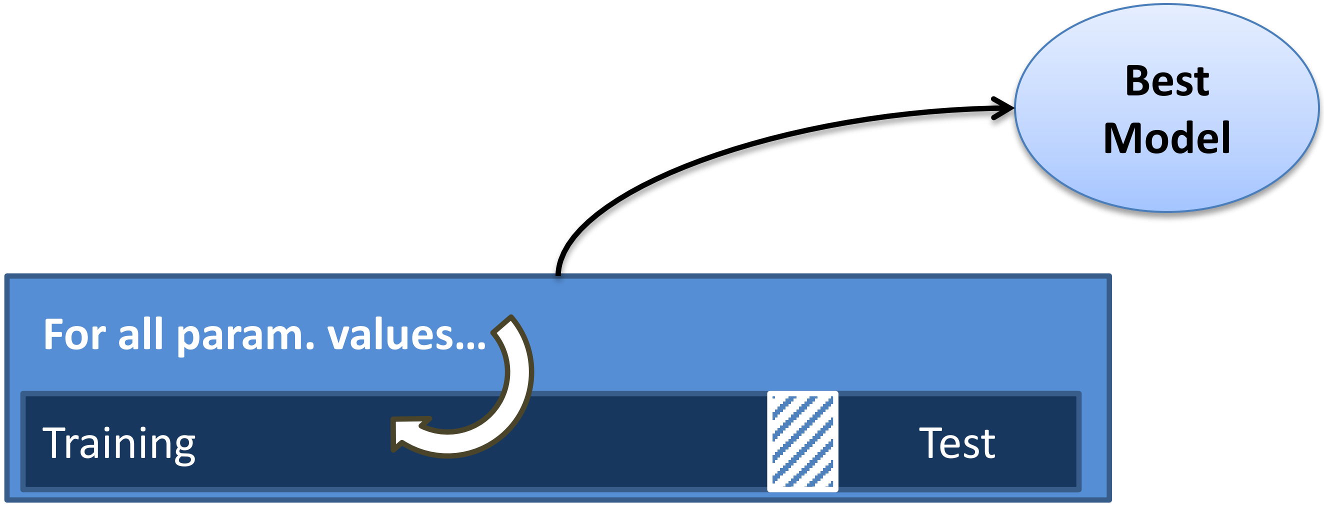
Evaluation: Offline

- Evaluation of computational approaches on a single data set?
- can split data set repeatedly into training/test blocks systematically, a.k.a. *cross-validation*



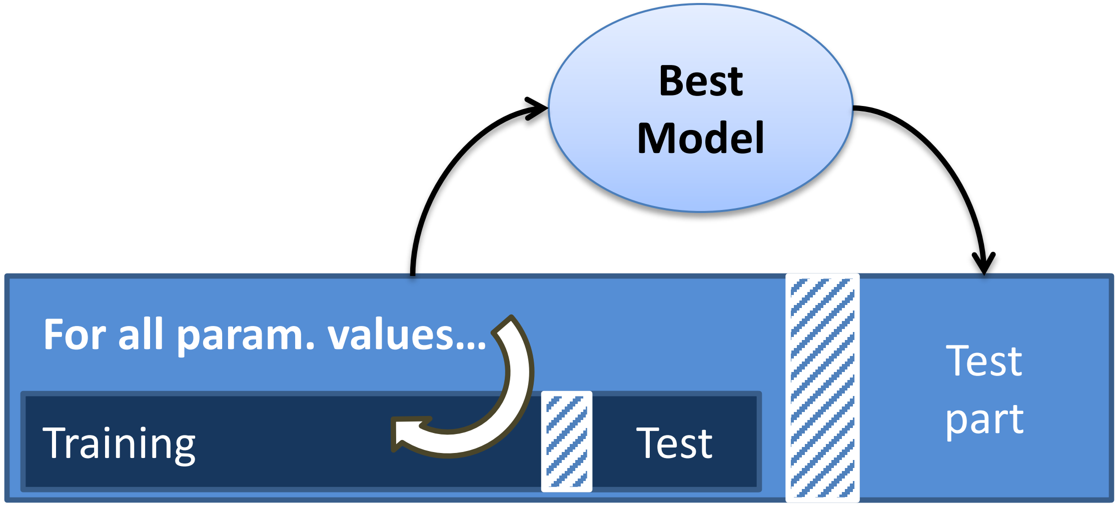
Finding Best Parameters

- Can be done using cross-validation in a grid search (try all values of free parameters)
- Quite general (e.g. can search for best method)



Finding Best Parameters

- Can be done using cross-validation in a grid search (try all values of free parameters)
- Evaluation: Can be nested *within* an outer cross-validation ("nested cross-validation")

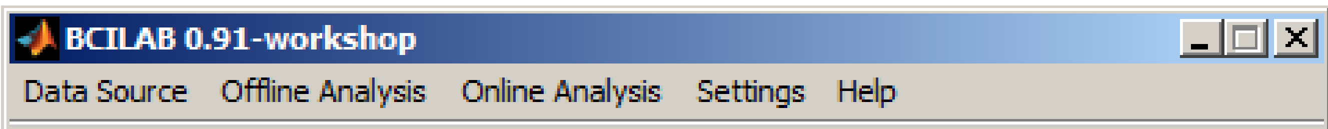


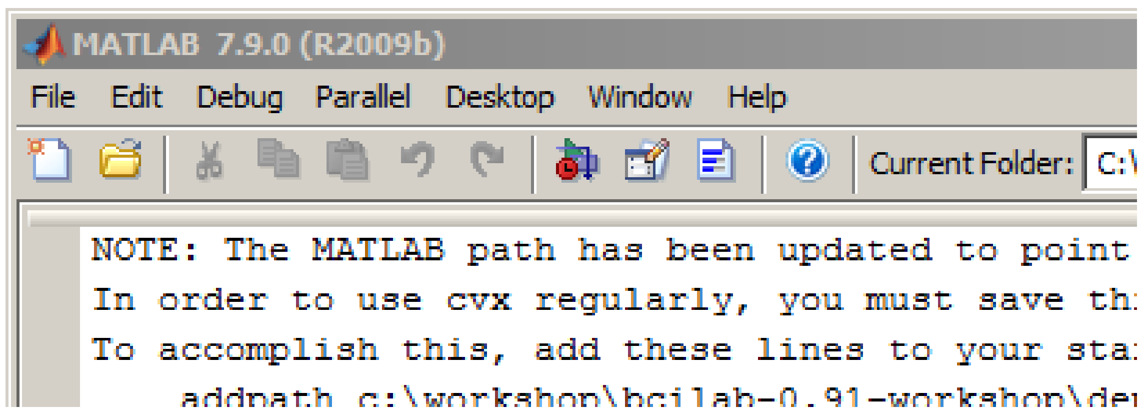
Neat Feature

- All offline analysis can be executed in parallel on a cluster (or the cloud)
- Batch analysis, cross-validation, parameter search, methods comparisons

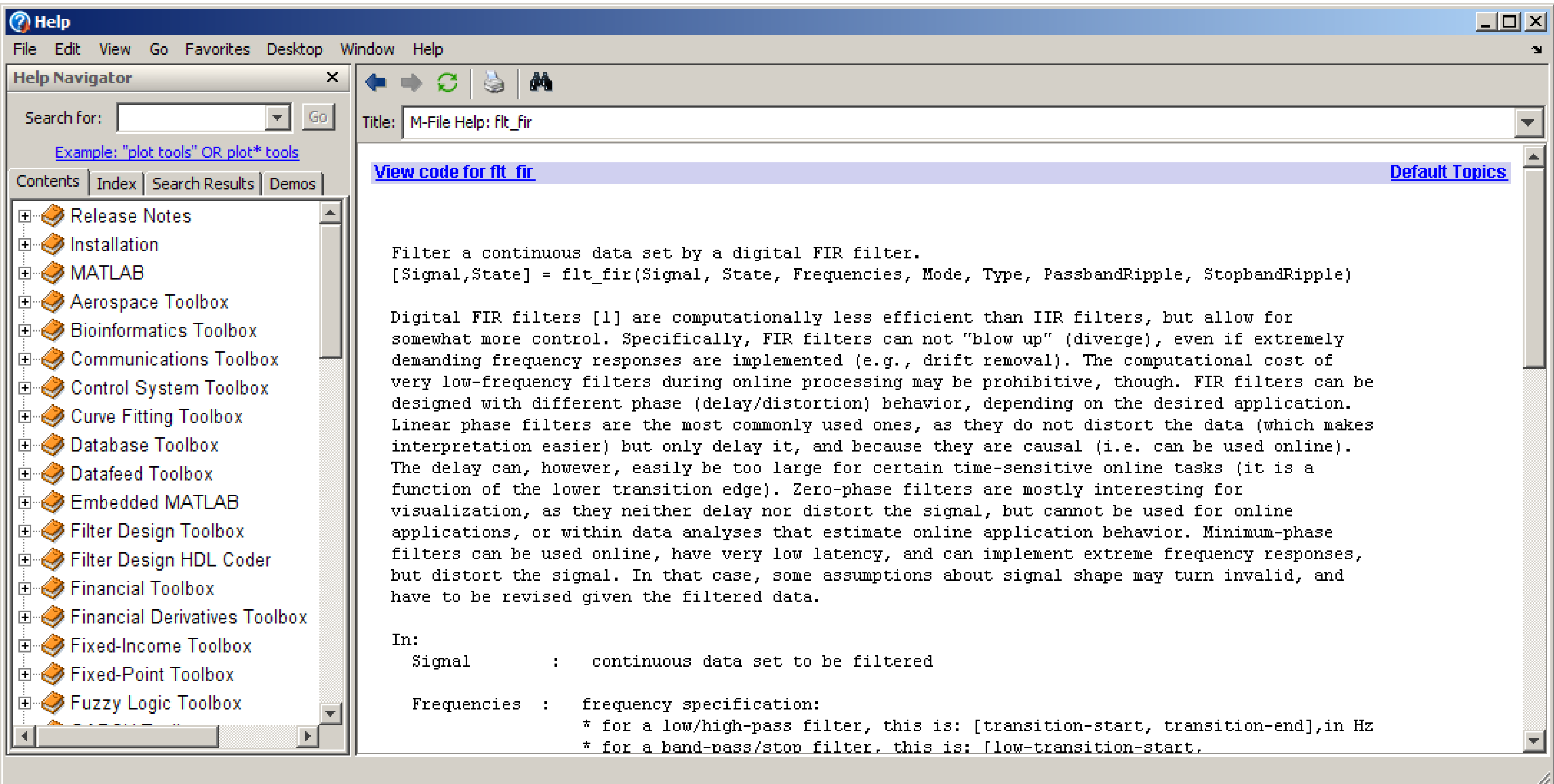
GUI Tour

- Alternative to scripting, for experimenters, psychologists, quick-and-dirty analysis

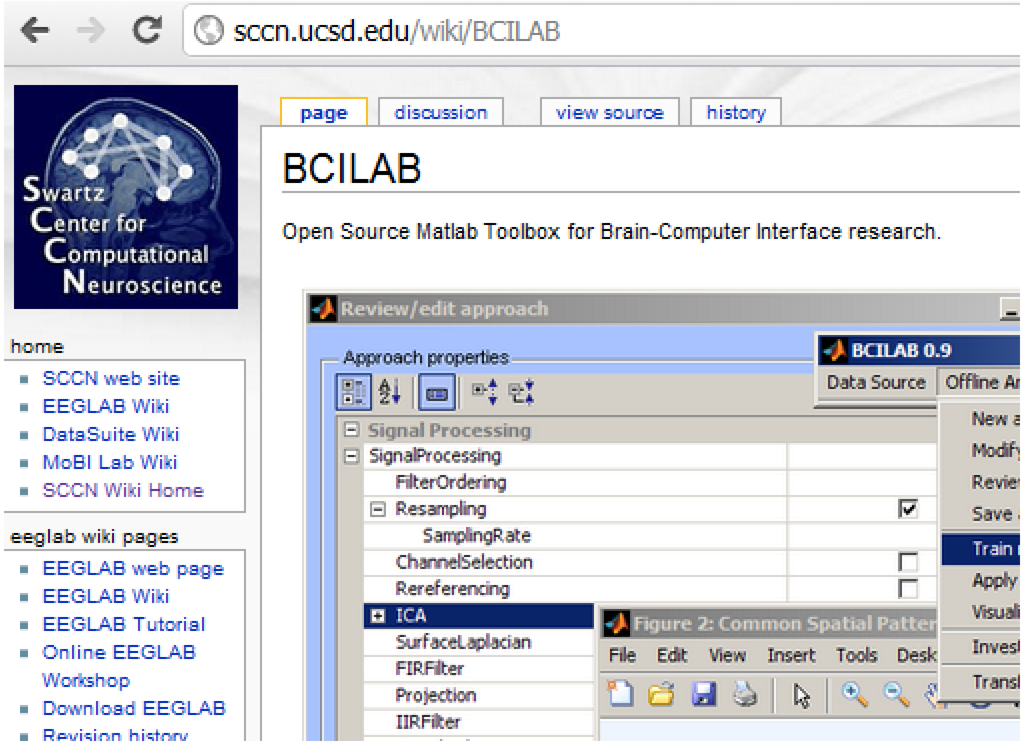




Integrated Help

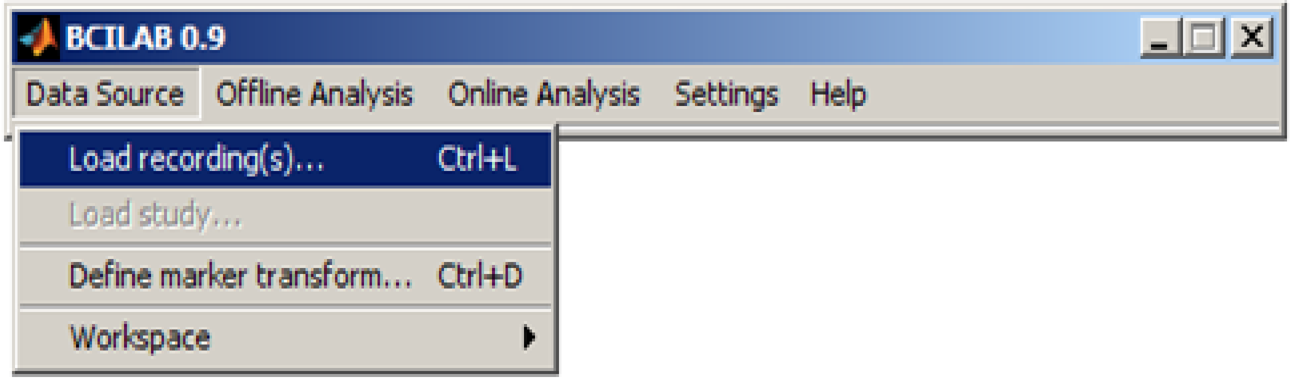


Help Wiki



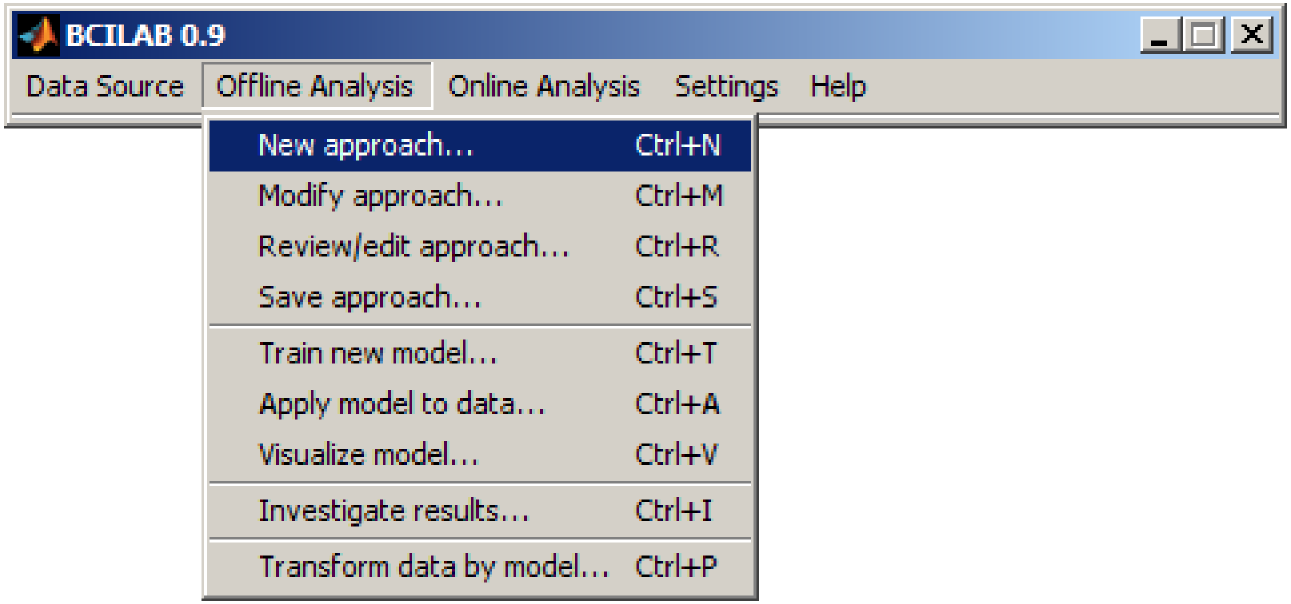
http://sccn.ucsd.edu/wiki/BCILAB

Loading data



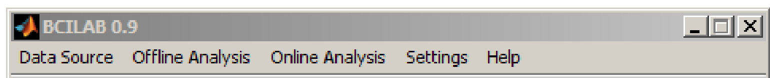
- Supports >20 common file formats

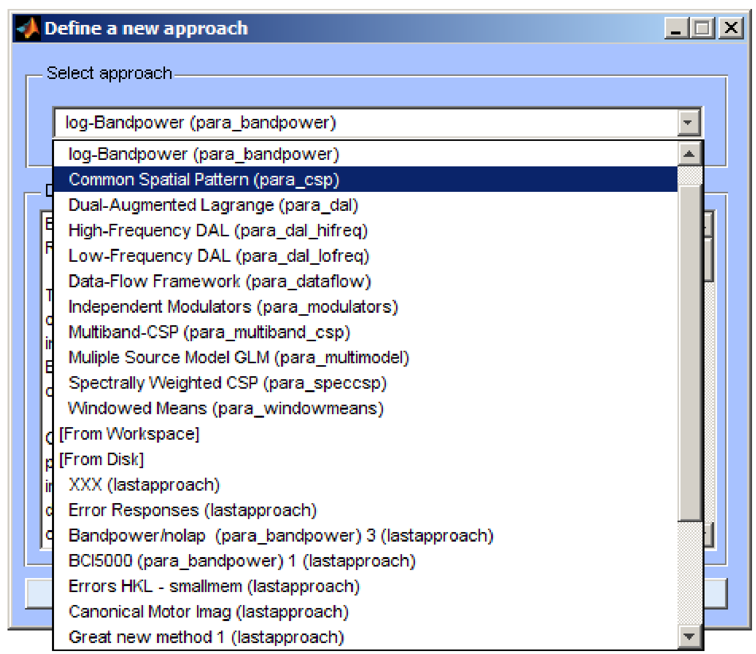
Offline Analysis



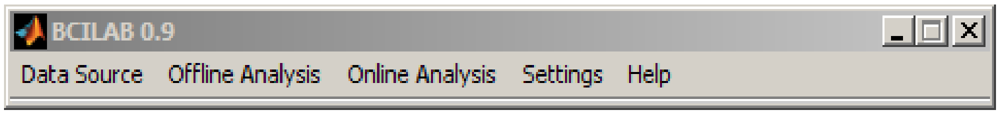
Defining an Approach

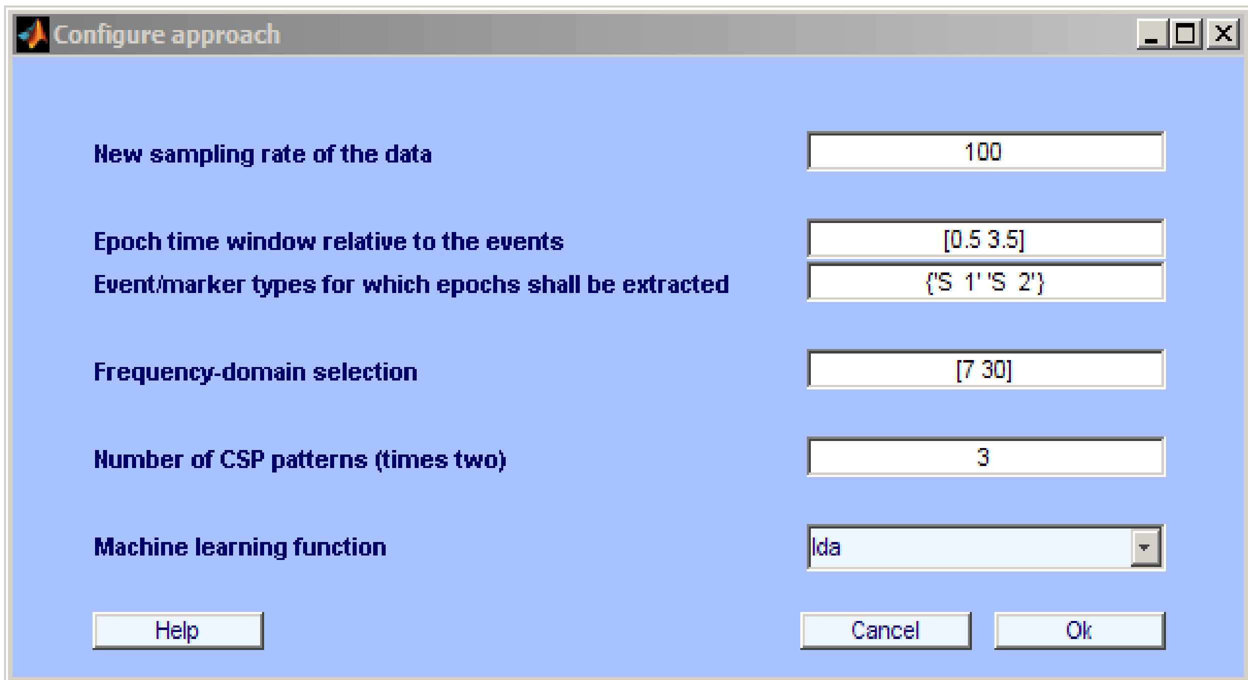
- Some of these work best for oscillatory processes, others for ERP-like features, etc.





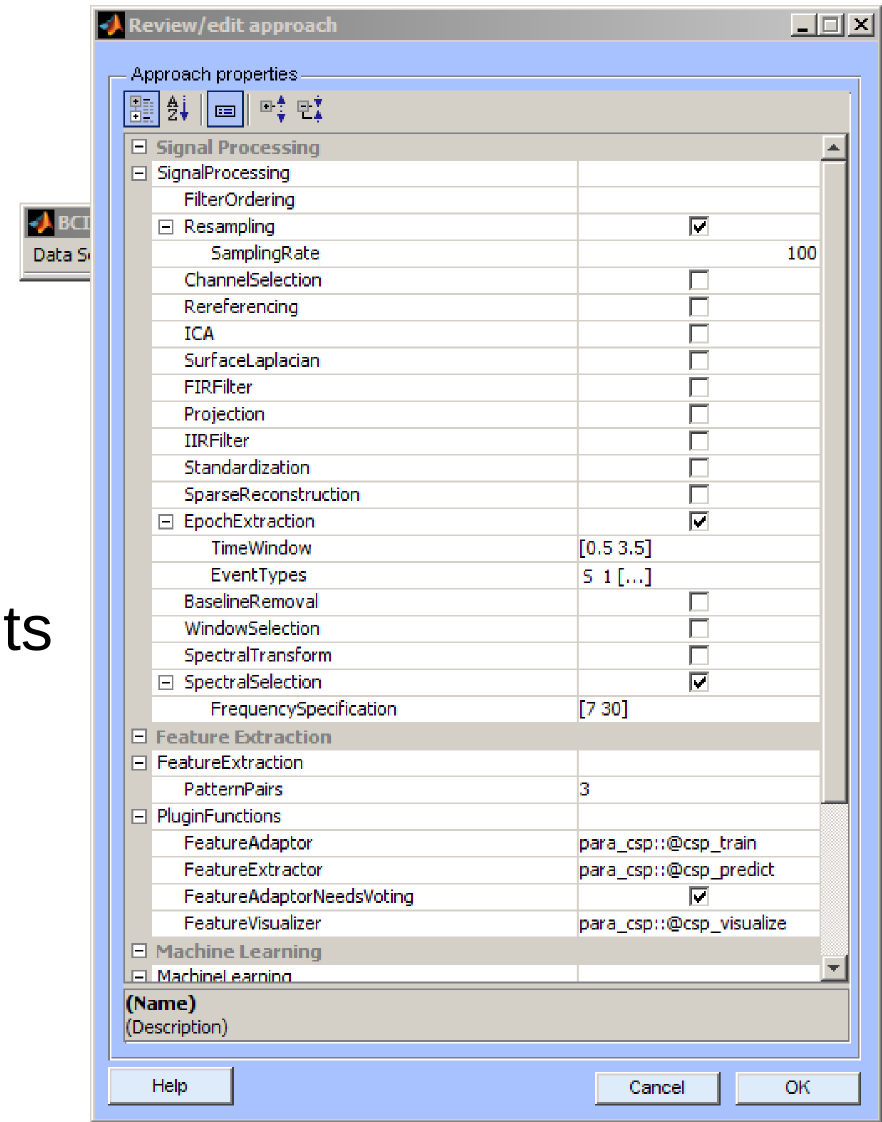
Quick Setup


BCILAB 0.9
Data Source
Offline Analysis
Online Analysis
Settings
Help

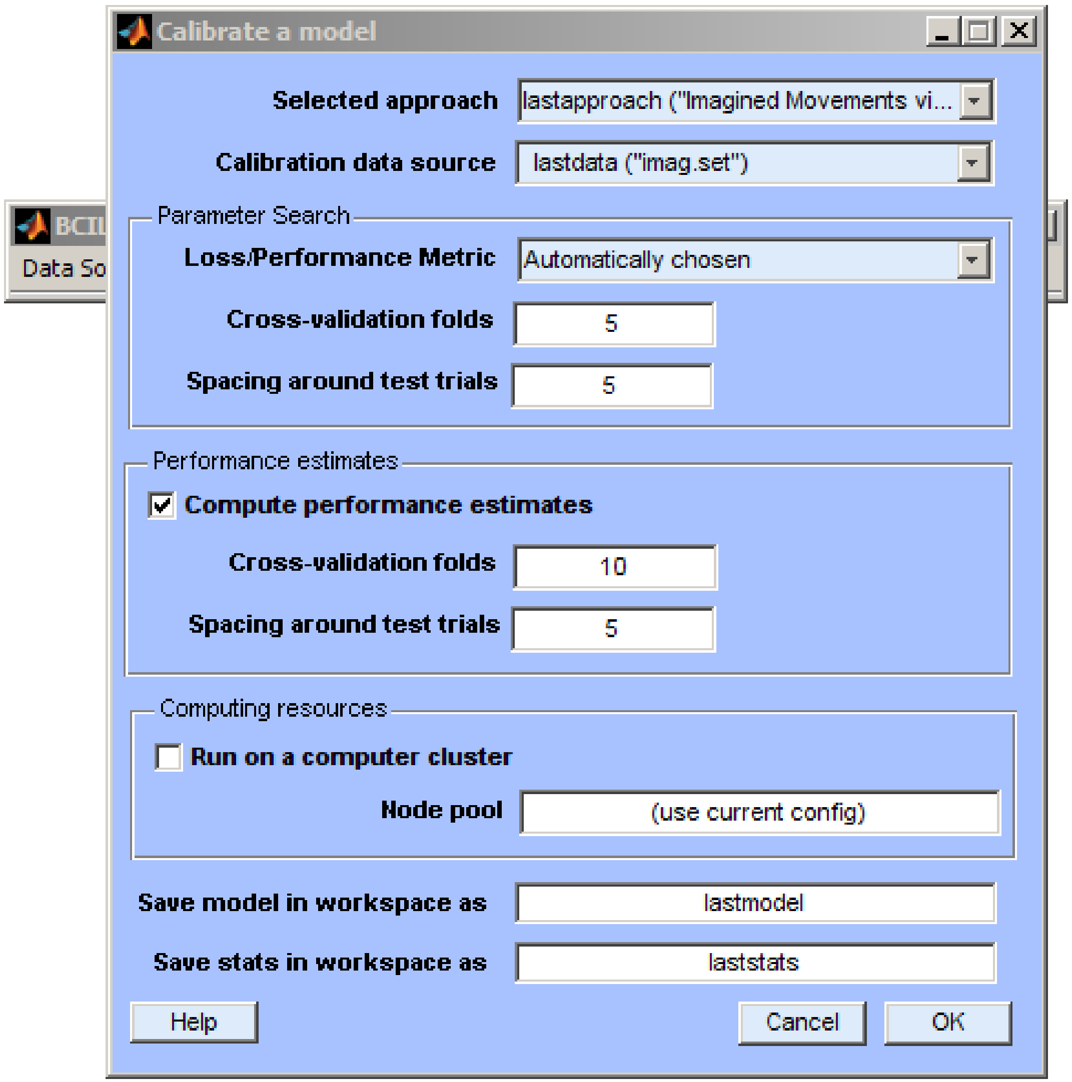

Configure approach
New sampling rate of the data
100
Epoch time window relative to the events
[0.5 3.5]
Event/marker types for which epochs shall be extracted
{'S 1''S 2'}
Frequency-domain selection
[7 30]
Number of CSP patterns (times two)
3
Machine learning function
lda
Help
Cancel
Ok

Or Detailed Configuration

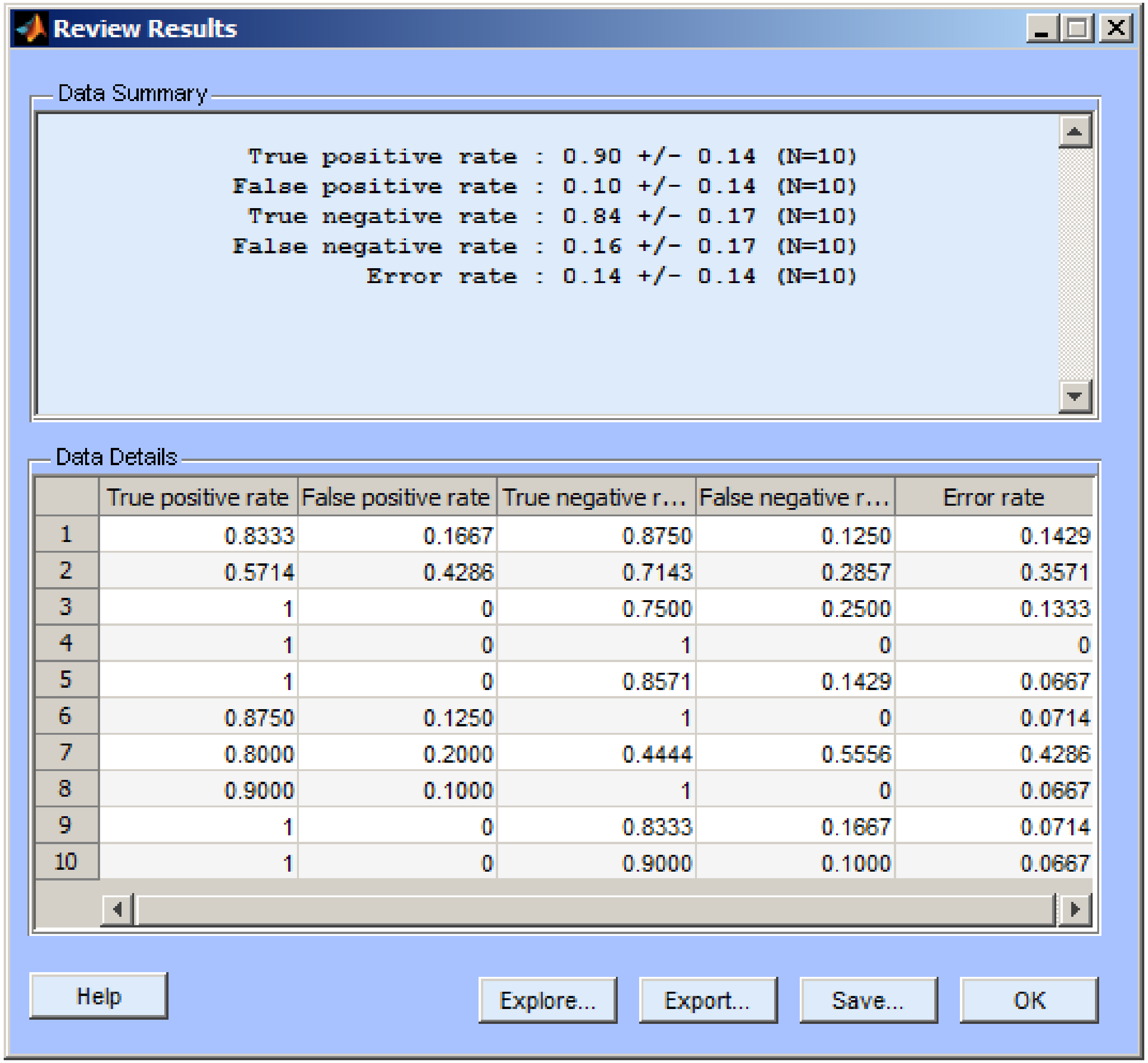
- Allows to edit all properties of the chosen approach
- Filter stages can be added and configured
- Feature extraction can be configured
- Machine learning components can be selected and configured
- …



Model Calibration



Reviewing Results



Review Results

Data Summary

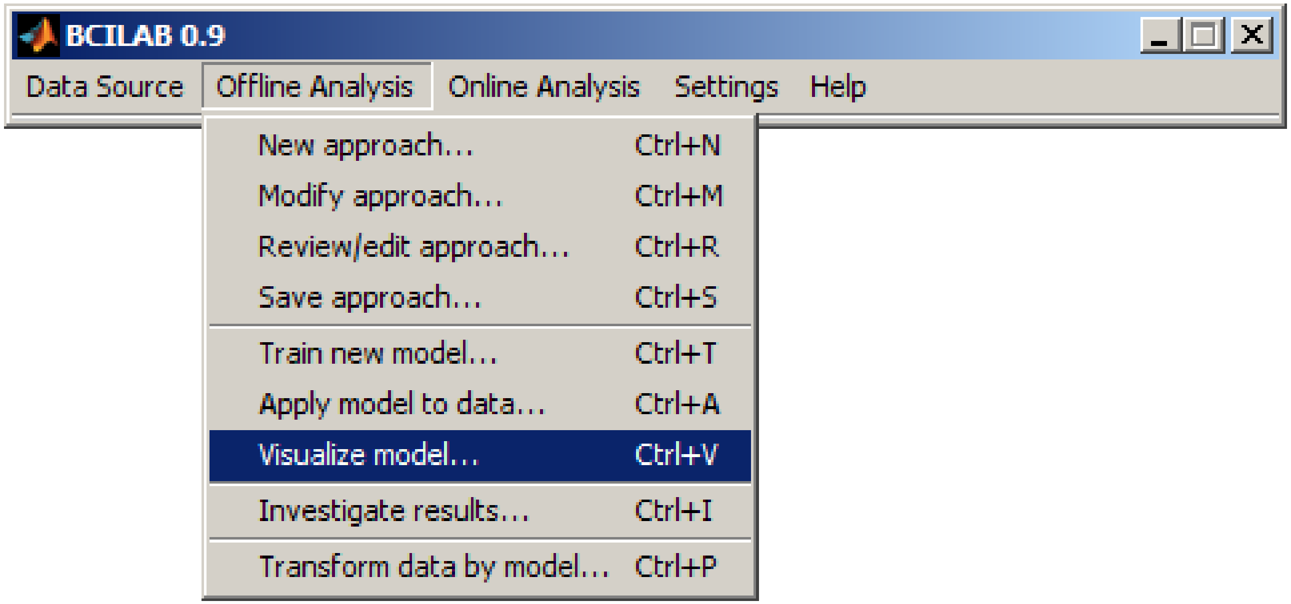
```
   True positive rate : 0.90 +/- 0.14 (N=10)
  False positive rate : 0.10 +/- 0.14 (N=10)
   True negative rate : 0.84 +/- 0.17 (N=10)
  False negative rate : 0.16 +/- 0.17 (N=10)
           Error rate : 0.14 +/- 0.14 (N=10)
```

Data Details

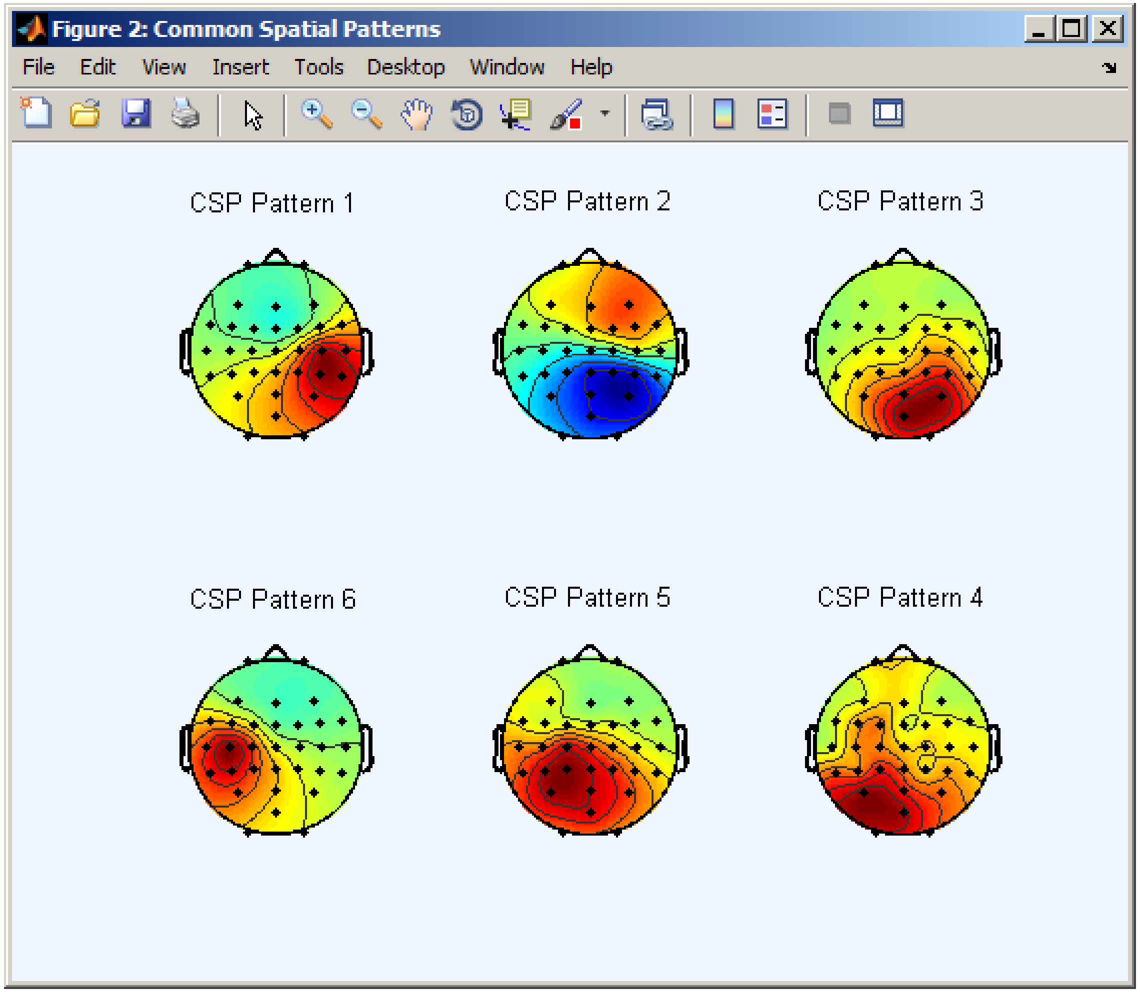
	True positive rate	False positive rate	True negative r...	False negative r...	Error rate
1	0.8333	0.1667	0.8750	0.1250	0.1429
2	0.5714	0.4286	0.7143	0.2857	0.3571
3	1	0	0.7500	0.2500	0.1333
4	1	0	1	0	0
5	1	0	0.8571	0.1429	0.0667
6	0.8750	0.1250	1	0	0.0714
7	0.8000	0.2000	0.4444	0.5556	0.4286
8	0.9000	0.1000	1	0	0.0667
9	1	0	0.8333	0.1667	0.0714
10	1	0	0.9000	0.1000	0.0667

Help | Explore... | Export... | Save... | OK

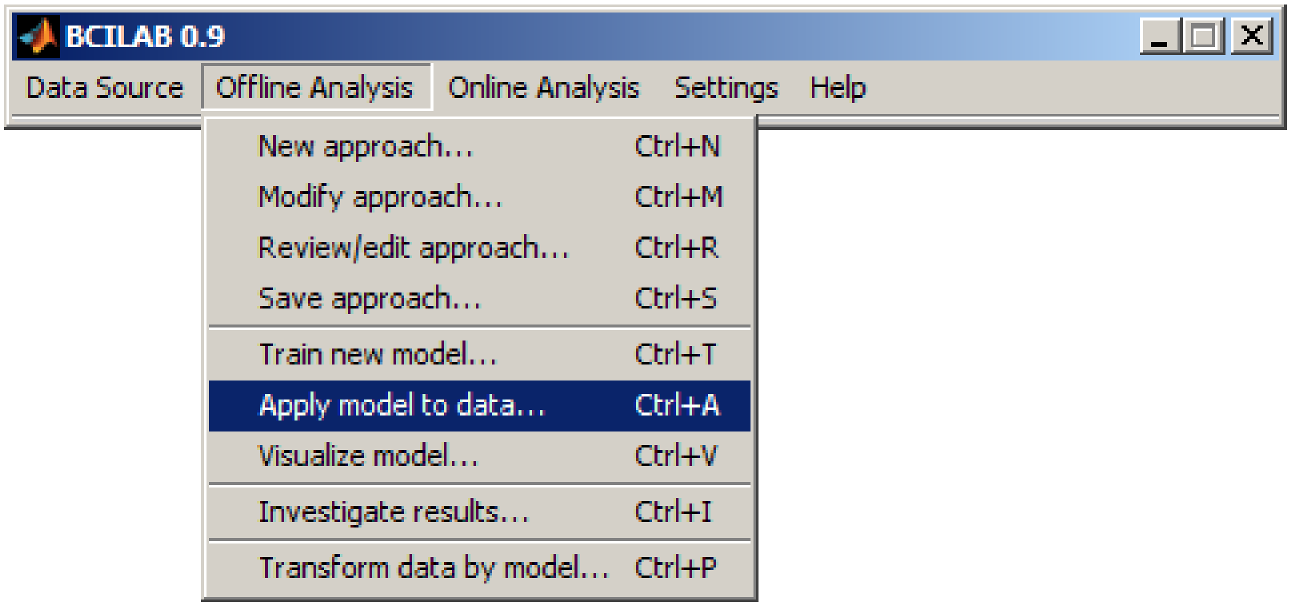
Visualizing model properties



Visualizing model properties

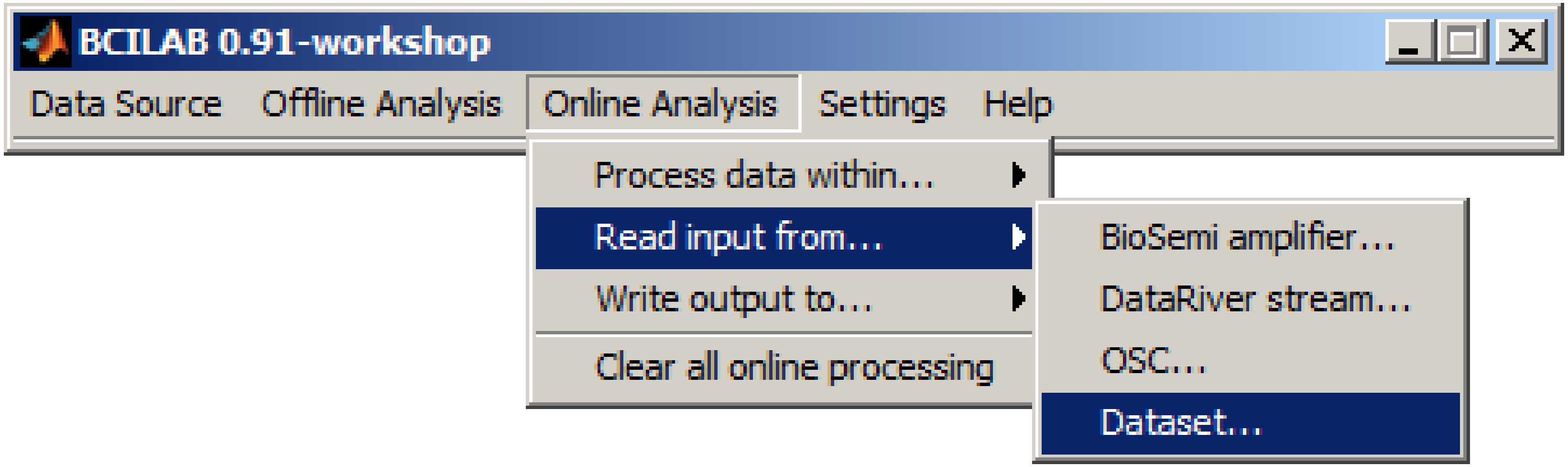


Applying Models Offline



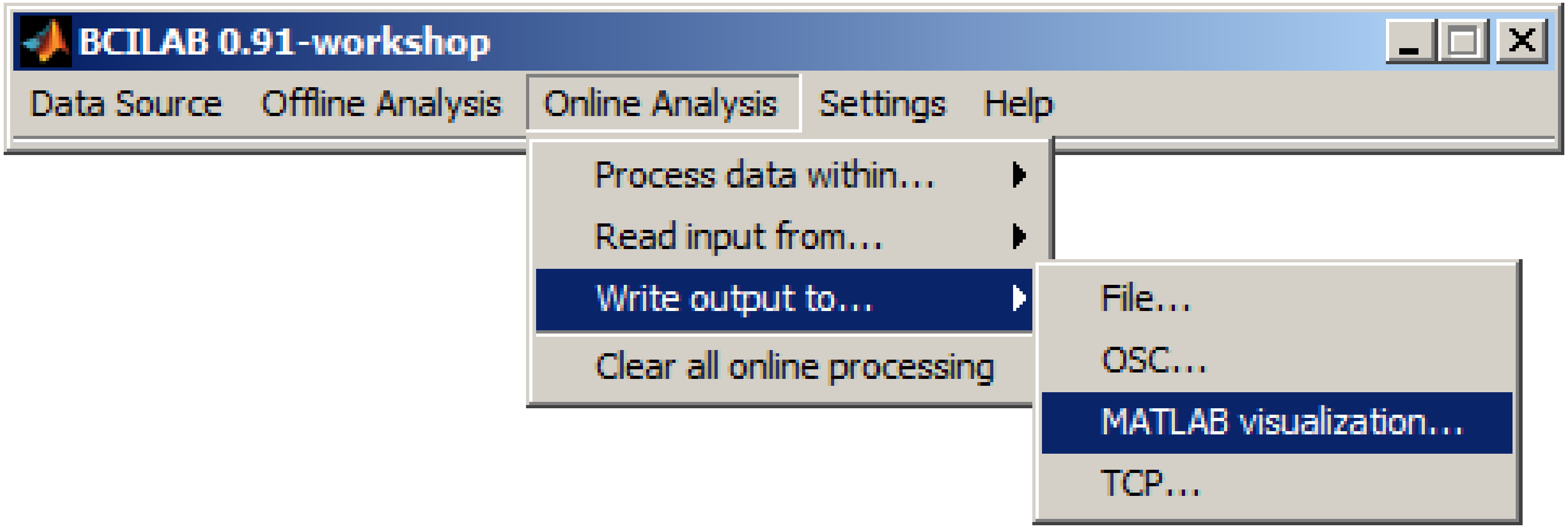
Online Processing

- Select data source

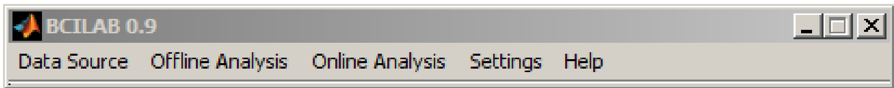


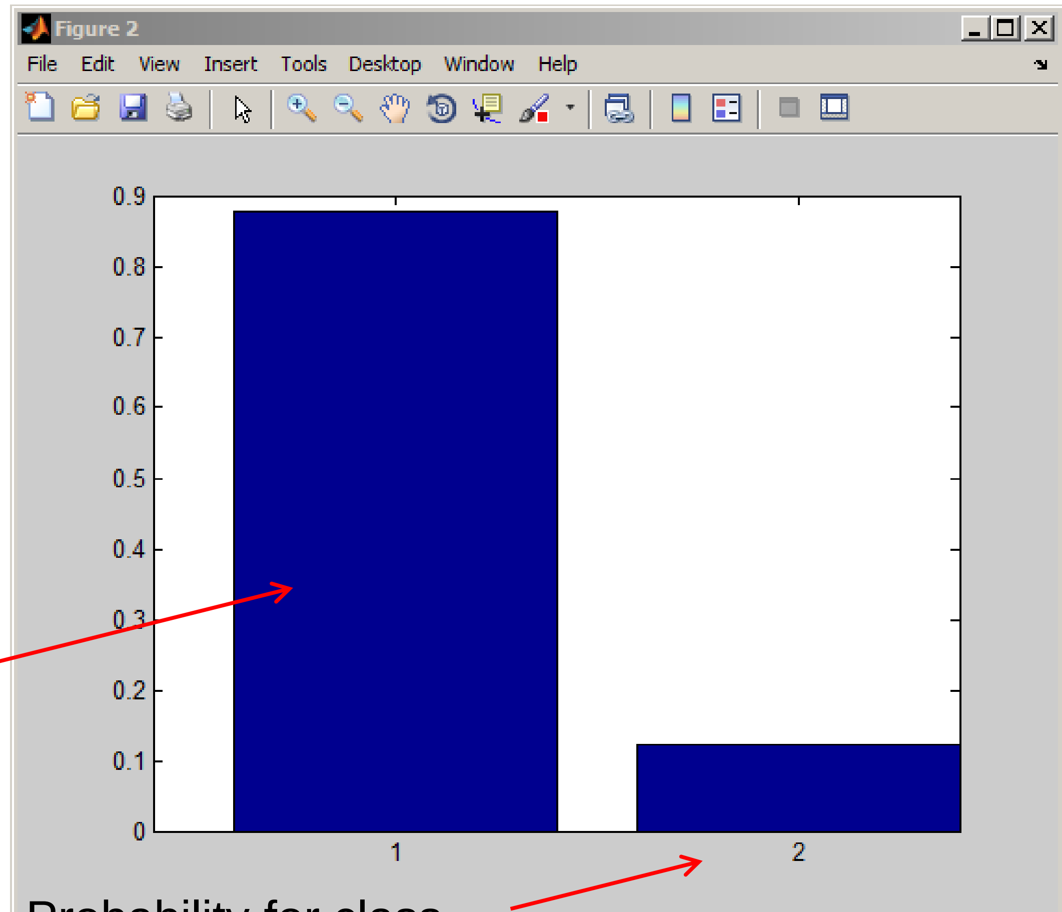
Online Processing

- Select output destination



Sample real-time output





Probability for class

Probability for class 2

Scripting Examples

- Doing a parameter search and nested cross-validation

```
%% --- train an alternative model with parameter search ---
% (over possible values for the number of pattern pairs, using CSP; note: this takes quite some time!)
% (the number of pattern pairs found optimal should be 3 in this case)

% load the data set (BCI2000 format)
traindata = io_loadset('data:/tutorial/imag_movements1/calib/DanielS001R01.dat');

% define approach
myapproach = {'CSP' ...
    'SignalProcessing',{'EpochExtraction',{'TimeWindow',[0 3.5], ...
                        'EventTypes',{'StimulusCode_2','StimulusCode_3'}}}, ...
    'FeatureExtraction',{'PatternPairs',search(1,2,3)}};

% learn model; here, using only a 5x outer cross-validation as it is otherwise too slow
[trainloss,lastmodel,laststats] = bci_train({'data',traindata,'approach',myapproach, ...
    'eval_scheme',{'chron',5,5}});

% visualize results
bci_visualize(lastmodel);
```

Search over different alternatives

Also: Custom cross-validation scheme

Scripting Examples

- Running the model online

```
% 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)');
```

Scripting Examples

- Running an 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
myapproach = {'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)
```

Scripting Examples

- Running a batch analysis for 3 modern approaches and 136 data sets (latest 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);
```

Plugin Authoring: 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:
%   Signal :   Epoched data set to be processed
%
%   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)

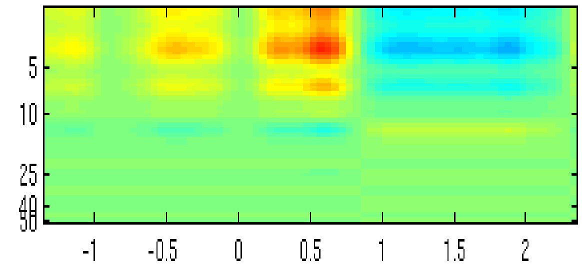
```
arg_define([0 3],varargin, ...
    arg_norep('trials'), ...
    arg_norep('targets'), ...
    arg({'cost','Cost'}, search(2.^(-5:2:15)), [], 'Regularization parameter. Reasonable range: 2.^(-5:2:15), greater is stronger. By default, it is average
    arg({'ptype','Type'}, 'classification', {'classification','regression','ranking'}, '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(g)
    g = 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
    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;
end
```

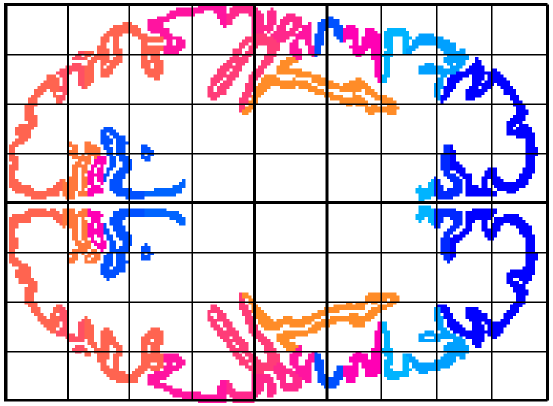
Complete CSP Paradigm

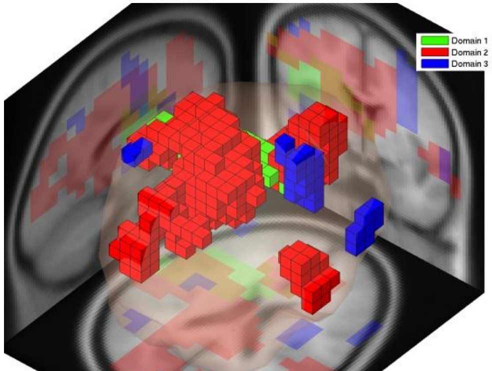
```
classdef ParadigmCSP < ParadigmDataflowSimplified
    methods

        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
                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: Source-Space Modeling

- Structural prior knowledge
  - can be introduced as side assumptions in the model (e.g. smoothness, sparsity, group sparsity, low rank, …)
- Quantitative prior knowledge
  - Structure atlases (Talairach, LONI, …) can supply information about the *a priori* relevance of a brain process
  - Can adapt the per-parameter penalty
- Empirical data
  - Data collected from other subjects can be co-registered/aligned and yield empirical prior distributions







Upcoming

- Next big toolbox release
- Online motion capture etc. processing via MOBILAB
- Effective connectivity integration via SIFT
- New methods in the pipeline (wave propagation imaging, alignment learning)

Thanks!

Questions?