1. Lecture 6.1 – Context and Background
   1. The BCILAB Toolbox
      1. Found on the interwebs
   2. Context
      1. Like EEGLAB, but for BCI (and/or cognitive state assessment)
         1. Seeding a community
         2. Strengthening links between BCI and Neuroscience
         3. Most widely used offline EEG analysis toolbox
      2. SCCN’s in-house tool for BCI problems
         1. Main Focus: Advance cognitive monitoring
         2. Part of a large US research program (CaN CTA)
         3. Funded by ARL (and ONR, Swartz Foundation, …)
      3. Toolbox is not just for sick people, it is for healthy people (like me)
   3. Software Environment For:
      1. Brain-Computer Interface Design (cognitive monitoring)
      2. Methods Research:
         1. Design & rapid prototyping of new methods & methods from literature
         2. Offline testing, performance evaluation & batch comparison
            1. Do lots of comparisons of different methods
         3. Simulated online testing
      3. Rapid Prototyping
         1. Real-time use
         2. Prototype deployment
   4. Facts & Figures
      1. Developed since 2010 at SCCN, UCSD (primarily by Kothe)
         1. Some help from others
         2. Completely rewriten
      2. Precursor was the PhyPA toolbox (Kothe & Zander, 2006-2009)
         1. Started in Berlin
      3. Built on top of EEGLAB (Delorme & Makeig, 2004)
      4. The largest open-source BCI toolbox by methods and algorithms (2012)
         1. Over 100 as of 2012
      5. Offline and online processing both in MATLAB, same code base, cross-platform, 32/64 bit
         1. Windows, linux, mac
         2. Can use on servers
         3. Compiled versions available so you do not need a MATLAB license
   5. Further Goals (these might have been accomplished)
      1. Provide large array of *existing methods* to reproduce existing literature – e.g., in benchmarking and comparison studies
         1. So you can say “ok what works best on my data”
      2. Provide *state-of-the-art and novel methods* to rapidly set up well-performing BCIs
      3. Provide plugin frameworks and backend solvers to implement new methods quickly
      4. GUI for beginners & experimenters, scripting for experts and MATLAB veterans – largely the same feature set
         1. I wonder how this is going…
      5. Allow for both conventional designs (e.g., data flow) and for radically new approaches
2. Lecture 6.2 – Quick Methods Teaser
   1. Time-Domain / ERP Baseline
      1. Windowed Means
         1. Traditional linear classifier for event-locked brain responses, usually using LDA
         2. Time windows manually assigned
         3. Examples: Error recognition, surprise
         4. For a given epoch, extract a bunch of segments, average the signal in these segments for each channel and utilize these features which are fed through a classifier which try’s to map that onto an output value
            1. Most frequently used method for BCI across literature
            2. There are things that are tunable

Classifiers

Parameters

* + 1. DAL-ERP
       1. State-of-the-art approach, no hand-tuned parameters
       2. Uses rank-regularized logistic or linear regression
       3. Complementary of “Windowed Means”, higher end
       4. No hand tunable methods
       5. Learns automatically, not only the spatial filter, also learns the time-course (the relevance over time), determines the number of these processes that are relevant
       6. Regularized and tuning parameter that is automatically optimized using cross-validation
    2. Oscillatory Processes Baseline
       1. Common Spatial Patterns Family
          1. Filter-Bank CSP (FBCSP)

Multiple bands/windows

Dominates this whole field

* + - * 1. Diagonal Loading CSP (DLCSP)

Cov. Shrinkage

* + - * 1. Composite CSP (CCSP)

Covariance prior

* + - * 1. Tikhonov-regularized CSP (TRCSP)

Filter shrinkage

* + - * 1. Spectrally weighted CSP (Spec-CSP)

Learning spectral filters from the data

* + - 1. DAL-OSC (higher-end)
         1. State-of-the-art approach, no hand-tuned parameters

Takes more time to train

* + - * 1. Also uses rank-regularized logistic or linear regression
        2. Single-step approach, jointly optimal
      1. If these do not find something interesting then there is probably not something interesting in this data
    1. New Methods
       1. Methods for Time-Domain Analysis
          1. Classify event-locked brain responses
          2. Best methods to date learn optimal evolving spatial filters
          3. Several methods in the same performance ballpark
          4. Examples

Error recognition

Surprise

* + - * 1. Benchmark paper in preparation
        2. Utilize all the information in a segment you decide
      1. Methods for Oscillatory Analysis
         1. Applicable to slowly-changing operator state and background activity as well as event-related transients
         2. RSSD is a pioneering method for learning full source-level time/frequency structure
         3. Examples: cognitive load, attention shifts
         4. Presented at ICON’11

Methods and data papers in preparation

* + - * 1. Takes 5-10 hours to process
    1. New Methods (Exploratory)
       1. Overcomplete Spectral Regression
          1. Long-term stationary oscillations
          2. Can integrate information from a corpus of data (across persons)
          3. Examples

Fatigue

Alertness

Sleep stages

* + - * 1. Presented at EMBC’11
        2. Related method presented at ABCI’11
      1. Spatio-Spectral Bayes
         1. A fully Bayesian version of RSSD aimed at neuroscientific modeling
         2. Allows for extensive statistical analysis of results
         3. Presented at Sloan-Swartz’11
    1. New Methods (Exploratory!!)
       1. Pattern Alignment Learning
          1. Finds time-jittered brain processes associated with known events in the work environment
          2. Radically new approach using joint optimization
          3. Applications

Target event detection and other event-related cognitive responses

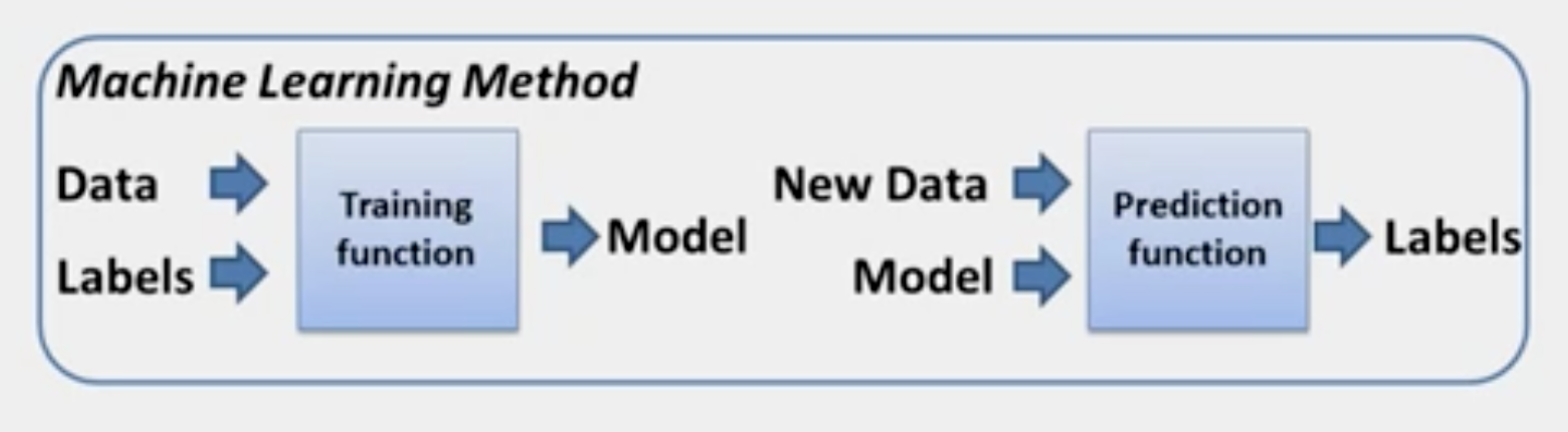
* + - 1. Independent Component Sparse Decomposition & Others
         1. General-purpose method for finding cortical source constellations that produce ongoing scalp signals
         2. Lifts restrictions on the number of source processes under consideration
         3. Using old code, needs to be updated
         4. Other methods

Auto-regressive modeling of joint human-system interface dynamics

1. Lecture 6.3 – Architecture Overview
   1. Toolbox Layers
      1. Multi-tier architecture
      2. Layer 3 – Framework [TOP LAYER]
         1. About
            1. The main internal clockwork of the toolbox that the user looks at
            2. Mechanisms that allow you to determine your approach
         2. GUI / Scripting Interfaces
            1. Approach Definition
            2. Outline Execution
            3. Offline Evaluation

I guess this is automated testing…

* + - * 1. Visualization
    1. Layer 2 – Plugins
       1. About
          1. The part that is extendable
          2. These are the where the plugins are implemented
       2. Signal Processing
          1. ICA
          2. SSA
          3. FIR
          4. IIR
          5. FFT
       3. Machine Learning
          1. LDA
          2. QDA
          3. DAL
          4. GMM
          5. SVM
       4. BCI Paradigms
          1. CSP
          2. Spec-CSP
          3. ERP
          4. RSSD
       5. Devices
          1. TCP
          2. LSL
          3. BCI2000
    2. Layer 1 – Infrastructure [Available for people who create methods]
       1. About
          1. This is for people who want to make GUIs or cache results or a bunch of helper functions such as things that deal with the operating system
       2. GUI Generation
       3. Cluster Computing
       4. Disk Caching
       5. Helper Functions
       6. Environment Services
    3. Layer 0 – Dependencies [EXTERNALLY MAINTAINED]
       1. CVX
       2. BNT
       3. EEGLAB
       4. GUI utils
       5. LIBSVM
       6. GLMNET
       7. Driver I/O
  1. Scope of the Online Framework
     1. You are supposed to write filters and prediction functions
     2. If you implement a new BCI you have to cram it into this framework
     3. Filter graph and prediction function
  2. Scope of the Offline Framework
     1. Uses pre-recorded data
     2. This is where the learning function is really made
     3. Also covered here is cross validation, grid search, nested cross-validation
        1. Used for seeing how good your BCI is working

1. Lecture 6.4 – Plugin Concepts
   1. Plugin Concepts: Filters
      1. Filters can operate on continuous signals
         1. Take a signal and turn it into another signal
         2. Most important type of filter
      2. Or on segmented (“epoched”) signals
         1. Take a collection of equal length segments and output the same number of segments of equal length
      3. *Static (“stateless”) filters*:  
         EEG = flt\_selchans(EEG,{‘C3’,’C4’,’Cz’})
         1. Does not transfer information from time to time
      4. *Dynamic (“stateful”) filters*:  
         [EEG,State] = flt\_resample(EEG,200,State)
         1. Does move data across time
         2. EEG is data set
         3. Can call on many chunks
      5. *Epoched filters*:  
         EEG = flt\_fourier(EEG)
         1. Take an epoched data signal and output one
      6. **Evil caveat**: filters have lazy evaluation behavior, i.e. they do not evaluate unless forced
         1. Don’t give you processed data, you need to force it to evaluate
         2. To force it  
            EEG = exp\_eval(flt\_fourier(EEG))
   2. Plugin Concept: Machine Learning
      1. Machine learning functions come in pairs:  
         
      2. Two functions…  
         M = ml\_trainlda(X,y)  
         p = ml\_predictlda(Xnew,M)
         1. X
            1. Matrix of obersvatiosn vectorized
         2. y
            1. labels
         3. Xnew
            1. New data
         4. M
            1. The model from the ml\_trainXXX() function
   3. Plugin Concepts: Paradigms
      1. BCI paradigms are the coarsest plugin type in BCILAB and tie all parts of a BCI approach together (signal processing, feature extraction, machine learning, …)
      2. They are invoked by the offline/online framework
      3. Simply a class in MATLAB
         1. Calibrate
         2. BCI Model
         3. Visualize (optional)
   4. Plugin Concepts: Online Readers
      1. Online reader plugins read signals from a source device and make them available in the MATLAB workspace
         1. E.g. BioSemi amplifier
         2. Example  
            run\_readbiosemi();
      2. Updates in the background
   5. Plugin Concepts: Online Writers
      1. Online writer plugins write BCI outputs (i.e., predictions) to some external destination
      2. Example:  
         run\_writetcp(‘mdl’,’strm’,’192.168.1.5’,12467)
2. Lecture 6.5 – Data Representations and Pipeline
   1. Data Representations
      1. BCI Model  
         Filter Graph  
         Predict
         1. Basically a struct
      2. Probability Distributions
      3. Feature Vectors
      4. Symbolic Expression
         1. Head  
            @flt\_fir
         2. Parts  
            { mydata, [0.5 1], ‘highpass’ }
      5. Signal – same as EEGLAB  
         data  
         -Matrix: number of channels By number of samples  
         3D matrix if you have segments   
         event  
         Struct array with events  
         srate  
         streaming rate  
         xmin  
         time of first data point  
         chanlocs  
         dipfit  
         … (meta-data)
      6. Signal Bundle  
         streams  
         n number of signals
      7. Dataset Collection  
         could be all the data you took in a day
   2. Pipeline Notation
      1. BCILAB is a framework that resembles a processing pipeline: first configure everything, then apply it to one or more data sets
         1. First configure a processing pipeline
            1. “I want to computer this and that”
         2. Then you actually apply it to data
            1. Apply it to calibration data and get a model
            2. Apply it to datasets and evaluate it
      2. Configuration Inputs:
         1. There are a couple key inputs that the pipeline needs to know before the pipeline can do anything
         2. You always need a dataset to crunch, which is just a time series
         3. Mapping between marker type strings and numeric class labels
            1. Any ML needs to know how marker strings like “Stimulus 1” relate to numerical class labels, the kinds of labels that the ML needs to predict
            2. This string relates to that number….
         4. Base BCI Paradigm to execute – “what to run?”
            1. What BCI paradigm you actually want to use
         5. Custom parameters for the paradigm
         6. Evaluation Scheme – “how to run it?”
            1. E.g. what type of cross-validation
      3. Pipeline Processes
         1. Curate
            1. Bring the input data into standard form
            2. Used to bring into a standardized form, add channel names, locations and so on…
         2. Design
            1. Define the computational approach
            2. What do I want to run, what parameters..
         3. Train
            1. Invoke all steps necessary for training (calibrating) a BCI and estimates performance
            2. Give dataset and get a model out
            3. By default will evaluate with cross-validation
         4. Predict
            1. Apply a BCI to some data offline
            2. Come with a new dataset and say what are the estimated labels
         5. Visualize
            1. Visualize BCI model Internals
         6. Run Online
            1. Apply a BCI online / incrementally
         7. Batch Analysis
            1. Perform a series of processing steps, optionally in parallel
            2. Large scale analysis of multiple methods and stuff
      4. Training Algorithm
         1. Train optimized model on entire data
            1. Optionally with parameter search
         2. Optional: do a cross-validation on entire data to quantify the model performance
            1. Optionally with nested parameter search
      5. A Note on Data Curation
         1. Up-front conversion of data set and file format idiosyncrasies into uniform representations:
            1. Continuous data – unfiltered, possibly reference
            2. Correct channel labels/locations

This could be a huge fucking problem.

* + - * 1. Correct event types, latencies, etc
        2. Other common meta-data about raw recordings
      1. Usually done in a first pass before any BCILAB function is touched