Data Mining Interface: A rough guide

This interface has been developed with the aim of making data analysis of any sensor data easy and online. It is a work in progress and contains bugs.

Basic Architecture

This is a Java web application built in Netbeans and exportable to a WAR file. The back-end is Java 1.7, the front-end is Javascript/HTML/CSS (with many libraries), and they communicate via Stripes (a Java interfacing framework) and jQuery posts.

Data input

Data must be put in .csv file format. Each row is a set of readings at a common time with a set of labels describing them.

ChanA, ChanB,ChanC, LabelA, LabelB
2.0000,3.0000,1.4000, Easycon,Malesub
2.200,3.2000,1.9000, Easycon,Malesub
.....

When consecutive readings have the same label, they are treated as one trial. When the label changes, it is treated as a new trial

INTERFACE

The front-end consists of three components (see Fig 1):

- **Data-object + Technique Area (top-left)**
  - All loaded datasets and instantiated data mining techniques appear here
- **Console (top-right)**
  - The user's primary method for input
- **Visualization Area (bottom)**
  - Visualization of data and stats

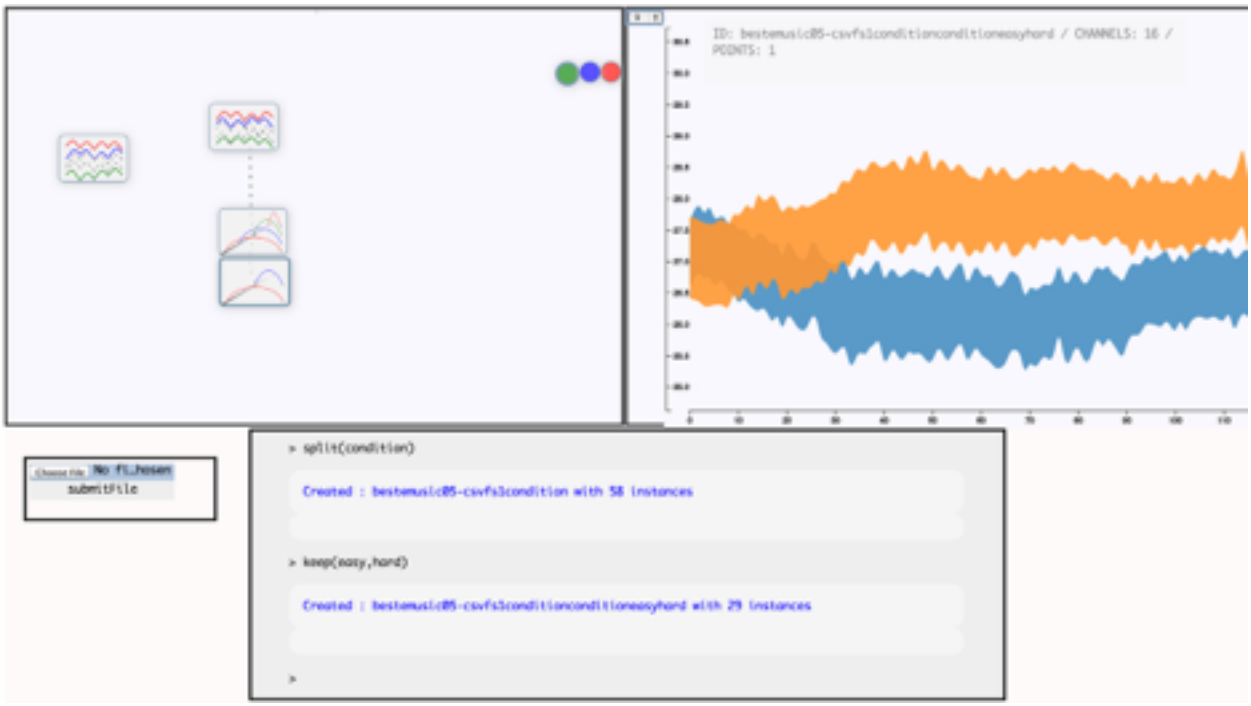
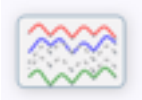


Figure 1. Interface

Dataset + Technique view

Every loaded dataset appears in the dataset view. Every-time the loaded dataset is filtered or processed, the new derived dataset appears next to it, connected with dashed lines. Datasets can be dragged around the screen for control of layout and to connect them to analysis techniques. If a dataset is single-clicked, a small set of metadata appears in the visualization screen. If it is double-clicked, the full dataset is visualized. Holding shift selects multiple layers at once, making their border orange and making so that commands apply to all of them. There are two basic views of the datasets and each exposes different back-end commands and visualizations:

2D Dataset



This is the raw form of data as it is loaded - just a timeseries of the readings at different channels. The fact that different readings pertain to different conditions is not relevant at this point. The data object maintains this form when filters and other data-manipulation techniques are applied to it (derived values connected by dashed lines appear next to it), but transforms into the 3D object when it is told what condition is currently relevant with the *split* command.

2D-datasets can be filtered and manipulated with the commands filter.movingaverage(), filter.lowpass(cutoff), filter.highpass(cutoff), filter.bandpass(low,high).

3D Dataset

Once a split command has been applied to a 2D object, it is transformed into a 3D object, enabling a new commands and visualizations. Under this representation, data is divided into trials with readings that share a condition. To see how predictive the trial's data-points tend to be of the class (ie, evaluate the quality of the data), this object is dragged to intersect a full-set of techniques. The command *evaluate* then returns the overall classification accuracy of the machine learning model.

Techniques
A complete set of techniques consists of a feature-set, an attribute selection technique, and a machine-learning algorithm. When a data-object intersects with technique-objects, the data will be evaluated according to the technique-settings for every detected combination of machine learning, attribute selection, and feature-set.Proper views for these objects have not yet been create, so these placeholders are used for now. Default values for each of these technique-objects appear for each and new one's can be set with the commands makeml, makeas, and makefs.

Machine Learning

New machine learning algorithms are instantiated with the command makeml(algorithm), and governs what machine learning algorithm will be used to evaluate the data. For now, algorithm can be set to jrip, j48, lmt, nb, tnn, smo, simple, logistic, and adaboost, which correspond to weka classes integrated in the back-end.

Attribute Selection

New attribute selection algorithms are instantiated with the command makeas((cfs || info, numAttributes). The first parameter specifies what algorithm to use, cfs for Weka's computationally expensive cfs-subset algorithm and info for the cheaper information gain algorithm. The second parameter specifies how many attributes to leave. An error is given if this exceed the number of attributes extracted by the feature-set.

Feature Set

New feature-sets are instantiated with the command makefs(ID), where ID is simply a name to give the feature-set. By default it is empty, so users must next select it and sue the command addfeatures(statistic,channel,window) to populate it with features.

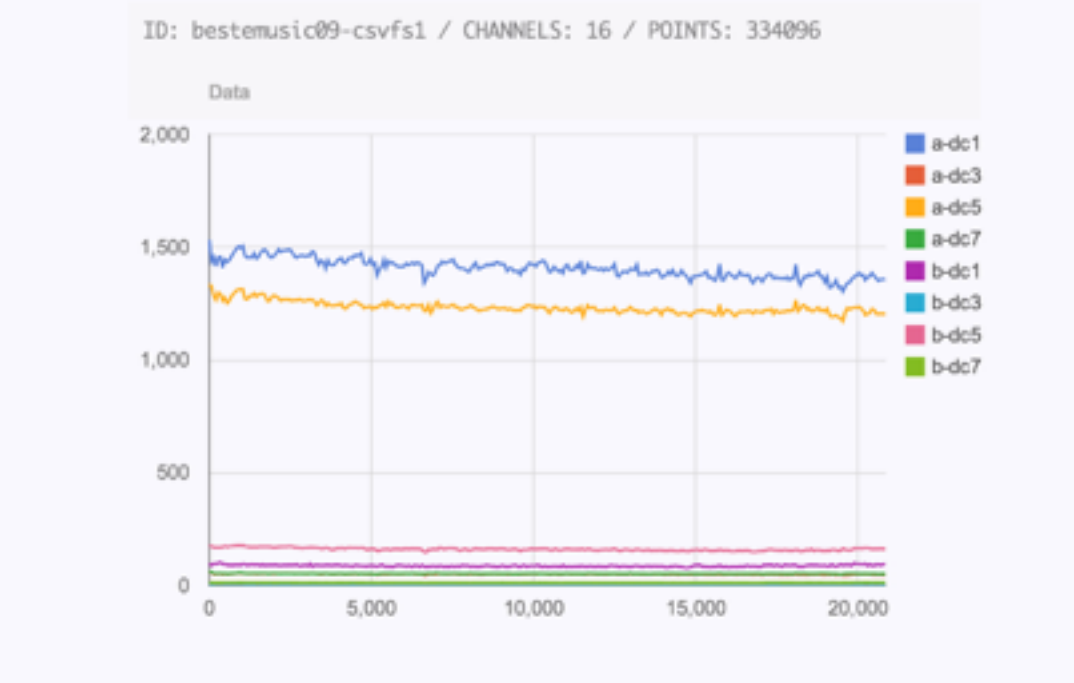
- *statistic*: mean, slope, secondder, fwhm, largest, smallest, absmean, t2p, and sax-xxxx (where xxxx that describes a canonical sax timeseries line-description from which line distances are computed)
- *channel*: referencable by id or index. 0^1^2 makes attributes for the first three channels
  - 0+1+2+3 averages the values at the first four channels, effectively creating a region
- *window*: what part of the instance: FIRSTHALF, SECONDHALF or WHOLE.
  - The parameter 6:10 would constrict the features to points between index 6 and 10

addfeatures(*,*,*) would add all possible combinations of features, except the one's that would result in an impossible quantity of features (like sax-xxxx in *statistic*, averaging channels in *channel*, and specifying precise windows in *window*.

Visualization View

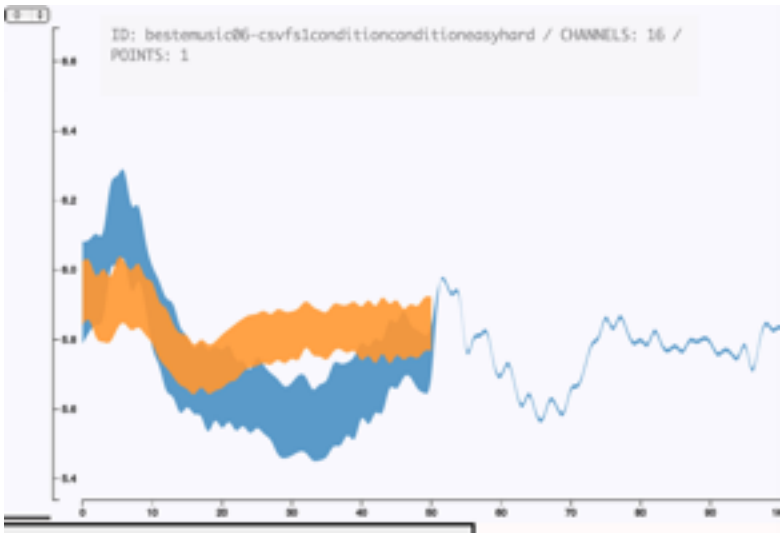
2D-datasets, 3D-datasets, and techniques all have visualization customized to their particular formats, memory-cheap meta-data from a single click and more memory-intensive data from a double click.

2D-data view



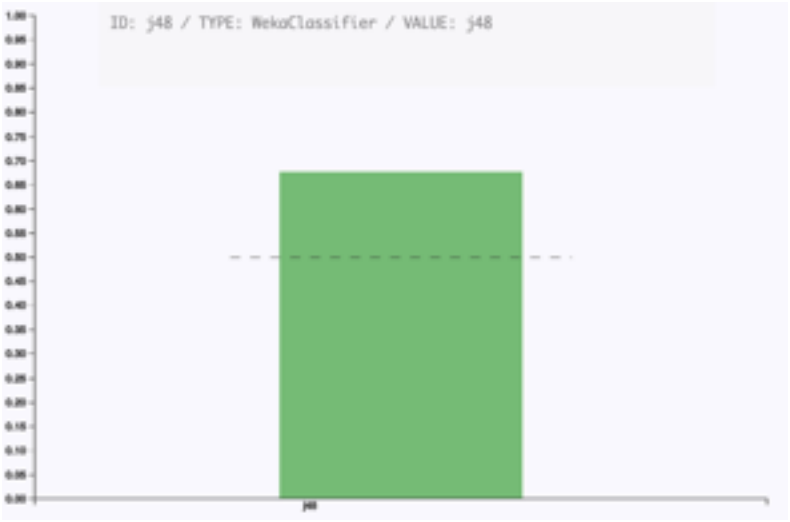
Built with the google chart API, this view shows the entire dataset at all the channels.

3D-data view



Built from scratch using D3, this animated chart shows averaged values in a trial in one channel under different conditions. In this case, there are two conditions. The chart transitions from a collection of individual lines that comprise the trials, colored by condition, into the thicker area chart.The thickness of the line is a function of standard deviation, so when the two areas depart from each other, it means that there is some information in the signal. To change channel, you switch the value on the top-left corner.

Performance view



Also animated and built from scratch in D3, the performance view shows the average classification accuracy with the selected technique or dataset. The dashed bar denotes where classification would have expected to have been by chance. The color of the bar is set by performance relative to chance, so that it is green above chance and red below chance. The opacity is also a function of this, so that 100% accuracy would be the clearest green, and 0% would be the clearest red. Clicking on the bar opens every individual test using that dataset or technique, so that the user can isolate the effect of the interaction between different techniques.

Console

The console provides an area for user's to enter to commands that instantiate new analysis techniques as well as filter, manipulate, and partition the data. Blue output communicates changes to the back-end component and the results of evaluation. Red output communicates errors made by the user or the system. In general, specific errors imply the user just passed an unrecognized command or parameter, entered bad input, or multi-selected datasets that are inconsistent with each other, from which the system can continue to proceed functionally. Less specific errors suggest the system has broken and might need to be restarted.