Preliminary documentation to the "mysort" package - a spike sorting package for Matlab

Felix Franke, BCCN Berlin
ff@cs.tu-berlin.de

February 16, 2011

1 Installation

Add the folder that contains the +mysort folder to your Matlab path. Your done. If you want to check if it works, run the file "E01_A.m" in the "examples" folder. You should see a figure and no warnings or errors.

2 Introduction

This is a preliminary version of a Matlab package that provides spike sorting algorithms, utility functions and visualisations. It has not been tested sufficiently yet and was only used with Matlab2008a and 2010a. The source code is not sufficiently documented, sorry for that. Also, not all the algorithms have been published. The papers are in preparation.

3 Principles of the software design

This implementation relies on object oriented programming (OOP). If you are not familiar with OOP, the source code might be hard to understand. However, the code was written in a way that everyone with basic Matlab skills should be able to use it.

4 Usability

Since this is a preliminary version of the documentation it only covers the basic usage of the package.

4.1 Optional function parameters

Most functions in this package allow you to specify optional parameters. This is done in a similar way as Python would handle it, or some native functions from Matlab: E.g.

the "plot" function of Matlab can be customised by providing name-value pairs. Instead of calling

```
plot([1:10])
```

you might e.g. call:

```
plot([1:10], 'linewidth', 3);
```

4.2 Basic usage of the sorter

Suppose you have your extracellular data in a workspace variable X with the rows of X being the individual recording channels (for multi electrodes). Furthermore, T contains the templates of your spike waveforms. The rows of T are individual templates. For multichannel data, a row of T contains the individual single channel templates concatenated. Then, the following steps will allow you to sort your data.

1. Set you Matlab path to contain the folder that contains the +mysort folder
2. Define the number of channels

   nC = size(X,1);
3. Define the length of your templates

   Tf = size(T,2)/nC;
4. Initialise a Bayes Optimal Online Spike Sorter

   botm = mysort.sorters.BOTM();
5. Tell the sorter what templates are present in the data

   botm.setTemplates(T,nC);
6. Provide the noise covariance matrix for your data

   % Warning: Dont do this to your real data. Use the correct
   % noise covariance matrix instead of the data covariance matrix
   R = mysort.util.mcDataCovariance(X, Tf);;
   botm.setNoiseCovarianceMatrix(R, nC);
7. Sort your data

   gdf = botm.sort(X);
8. Plot the sorting

   botm.plotLastChunkSorting('X',X);

   % Warning! This will only work for small X! Remove this line for longer X.
   botm.plotSorting('X',X);

A couple of warnings concerning this sorting:

1. The sorting will only work for data that fits into your working memory! For longer data, a slight modification to the use of the sorter has to be made, that is not fully covered in this documentation yet but see "About the data handling" below.

2. This will result in poor sorting, since the noise covariance matrix is set to the data covariance matrix.

3. This might result in poor sorting if the number of templates is not right.

4. This might result in poor sorting, since the parameters for the BOTM will be set to the default values which might not be optimal for your case. Try e.g.
   botm = mysort.sorters.BOTM('upsample', 1);
   which will decrease the performance but also the computation time by a factor of roughly 4.

4.3 Examples

If you want to learn more about the sorter, maybe a couple of simple example files may help. In the folder "examples" you will find the corresponding Matlab scripts. For every example, there is a readme.txt, that should explain what this example illustrates.

4.4 About the data handling

In the above example, it was assumed that you have a local variable holding the extracellular data (X). This is the easiest way to use the package. This will be referred to as the "local data mode" of the spike sorting package. However, it might be very inconvenient to load all data at once into the memory. Therefore, the package allows for what will be referred to as the "dataInferface mode". Here, instead of providing the data directly, you will have to provide an instance of a class that inherits from "mysort.datafile.DataFileInterface".

An example might be the "Robin" class which even exploits Matlabs capability to use memory maps that allow you to use memory on your hard drive as if it was loaded into the working memory. This allows you to sort arbitrary large data sets. You will set the sorter classes into the "dataInterface mode" by providing the 'datafile' parameter to the constructor:

botm = BOTM('datafile', D, ...);

4.5 Some functions and classes:

Class BOTM: This is the class that implements the Bayes optimal template matcher (BOTM) (publication in preparation).

4.5.1 Functions for plotting

- botm.plotTemplates(...)

  Plot the templates that are stored inside the bss class.

- botm.plotSorting(...) (”datafileInterface mode”)
  botm.plotSorting('X',X,...) (”local data mode”)

  Plots a piece of data, the detected spikes and their templates.

- botm.plotLastChunkSorting(...) (”datafileInterface mode”)
  botm.plotLastChunkSorting('X',X,...) (”local data mode”)

  Since the BOTM is implemented as an online spike sorter, it will ”forget” most of the internal variables, once a chunk of data is sorted. Thus, only the last chunk can be plotted e.g. with the corresponding filter output functions.

- botm.plotISI(...)

  This will plot the inter spike interval distributions of the sorted spike trains.