Brain Computer Interface for ALS
Tackling The Non-Stationarity in Brain Signals

Shadi Iskander and Wassim Saba, Supervised by Ben Kartzu

Introduction:

BCI maps brain signals into commands for external computer application. This is done by using machine learning to train a classifier to interpret the brain signals and classify them into classes. For example, left and right-hand movement. In order to achieve practical usage of BCI, the classifier needs to match the user's brain activity and obtain good accuracy rates throughout time.

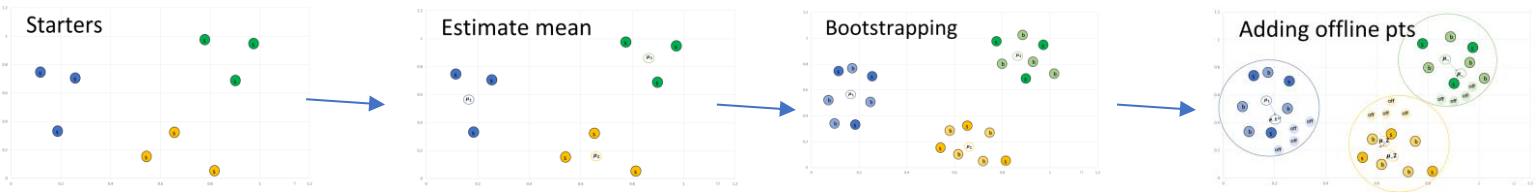
The non-stationarity of the brain signals and its instability makes it hard for the classifier; a classifier that has been trained on today's brain won't likely be able to achieve good accuracy rates on the brain signals of tomorrow.

Proposed Solution: Part 1

Our first proposed solution is composed of three steps:

- Offline: First, we will collect offline labelled data from few days. Each sample with d features extracted using feature extractor.
  For each feature F_i, we calculate the KL-divergence between each two days $D_{KL}(F_i^{day1}||F_i^{day2})$: if it's higher than KL threshold we keep it, otherwise we drop it.
  After feature filtering using KL-Divergence we continue to the next step with $k \ll d$ features

- Pre-online: this step includes partial calibration of the classifier in each day.
  Each morning we sample from the mentor few labelled samples. Applying bootstrapping to the new data. By the assumption that the data distribution changes with small motion we can leverage from the offline data, we add samples from the offline set that are bounded by radius R from the mean of each class in the new data, and this will be the train set for the specific day which the classifier will be trained on. The pre-online step can be interpreted as composing the train set out from old and new data, in a weighted way, the distribution of the train set will be similar to the offline data as much as the new data is similar to the offline data – this coincides with our demands/aim: if there is no data shift we want the train set to be as much the same as the offline data – because we have a lot of data there. But if there is drastic data shift – we won't be using offline data because it's irrelevant and we would have to work with the new samples only.

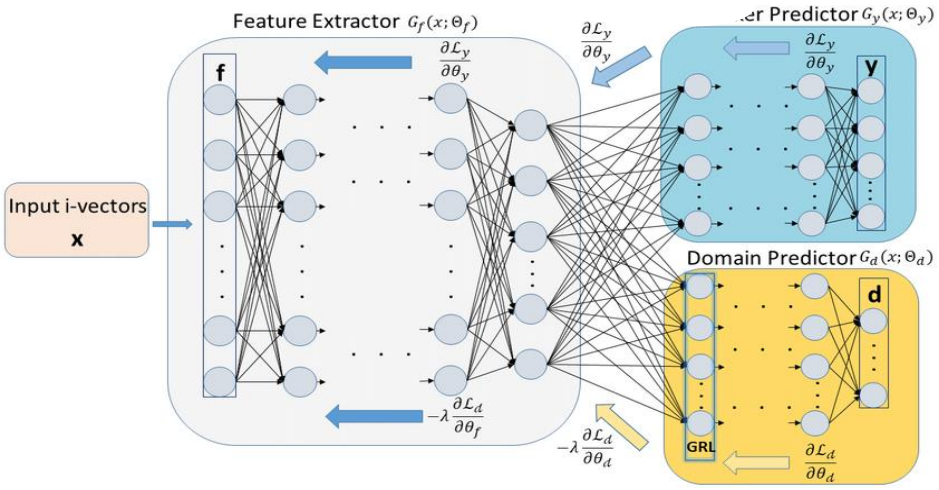


- Online: in the online phase we predict each sample using the pre-trained classifier from the pre-online
  For each sample with its prediction we calculate the Confidence Ratio : how much are we confident about the prediction. If it is beyond CR threshold – we add the sample along with its label to the train set, for each M new samples we fine-tune the classifier.

Proposed Solution: Part 2

In our second solution we attack the problem in a deep-learning approach:

We use the Domain Adversarial Training Network by Y. Ganin et al



The offline step here is the same as in Part1: we have labelled data collected from a few days.

We look at days as domains– in each day we are sampling from different domain distribution

Our aim is to learn features that are domain independent. Inspired by the theory on domain adaptation suggesting that, for effective domain transfer to be achieved, predictions must be made based on features that cannot discriminate between the training and test domains. How is this accomplished?

We have few fully connected layers which represent the feature extractor, which is connected to a classifier (another fully connected layer)

In addition to the classifier for label prediction – we add a domain classifier, which has to predict in which domain the sample is drawn from, however, instead of passing the gradient of domain loss $\frac{\partial L_d}{\partial \theta}$ to the feature extractor – we pass $-\frac{\lambda \partial L_d}{\partial \theta}$, aiming to **maximize** the feature extractor domain loss function rather than **minimizing** it, in the same time minimising the label loss function – in total to learn feature extractor generalized for all domains (days) – independently from which day we are recording.

Results

*Naïve algorithm: learns once from offline data and doesn't update itself (classifier doesn't change)

*Optimal algorithm: updates itself by doing full calibration every day (without addressing non-stationarity)