Tensor Decomposition for Gaits

Not to be distributed outside SPAN-PACER Lab

Injury Recovery Prediction

This repository performs injury recovery prediction based on features obtained from tensor decomposition method. Number of features are first selected by setting the acc_vs_samples to False. The obtained number of features is then used to test the setup for acc_vs_samples set to True. Tucker decomposition is used for feature generation while, MLP is used to learn non-linear approximation and kNN to form a baseline. Other classifiers used are SVM and Nearest Centroid.

Dataset This dataset is made up of three classes. Each class comprises of samples of a subject walking, during different stages of recovery from injury. The stages chosen for data were 10% recovered, 50% recovered and 95% recovered. The data consists of tri-axial accelerometer readings for the sensor mounted on heel of the subject. The data is organized in the form of tensor and each tensor is organized in the .mat file namely, recovered10.mat, recovered50.mat and recovered95.mat. The dataset has been re created. That is, the chopping procedure for each step has been re defined in accordance to the last meeting of Fall semester attended by Dr. Ivan Puchades, Tristan Scott and Kshitij Bichave.

Fig. 1: Recovered - 10% data. Subplot 1: X axis, subplot 2: Y axis, subplot 3: Z axis. On the x axis of each subplot data points. On the y axis of each plot is acceleration (g).

Fig. 2: Recovered - 50% data. Subplot 1: X axis, subplot 2: Y axis, subplot 3: Z axis. On the x axis of each subplot data points. On the y axis of each plot is acceleration (g).

Fig. 3: Recovered - 95% data. Subplot 1: X axis, subplot 2: Y axis, subplot 3: Z axis. On the x axis of each subplot data points. On the y axis of each plot is acceleration (g).

Method Run main.py to obtain results. To obtain results for acc_vs_reduced_dimension set acc_vs_samples = False. To obtain results for acc_vs_samples, set acc_vs_samples = True. The classifiers choosen are Multi Layer Perceptron(MLP) and k-Nearest Neighbors (kNN). MLP are best to obtain non linear approximation while using multiple hidden layers while the kNN nearest neighbors represent the model based on it distance to the k nearest neighbors.

Installing and Executing

Clone this repository

```
$ git clone https://github.com/kbichave/tensor-decomposition-for-gaits.git
$ cd tensor-decomposition-for-gaits
$ pip install -r requirements.txt
$ python main.py --samples # run this for acc_vs_samples
$ python main.py --dimension # run this for acc_vs_ReducedDimension
$ python main.py --all # run this for acc_vs_ReducedDimension_vs_Samples
```

Results

Fig. 4: Accuracys vs Reduced Dimension | Number of training samples per class: 20, Number of test samples per class: 4, Number of realizations: 500. MLP architecture: [solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(5, 2)]

Fig. 5: Accuracy vs Samples | Number of training samples increased from 0 to 30, Number of test samples: 4, reduced dimension l = 18, Number of realizations:500. MLP architecture:[solver='lbfgs', alpha=1e-5,hidden_layer_sizes=(5, 2)]

More on Classifiers:

- MLP on Wikipedia
- More about MLP execution on SkLearn
- Support Vector Machine
- k- Nearest Neighbour
- Nearest Centroid

[PM: Please specify the simulation details (k; crossvalidation method; number of realizations; reduced dimension; etc.)][PM: Present the classifiers used in more detail. E.g., how was the MLP implmented] [PM: Cite any papers needed for the methods used][PM: Present the structure of the tensor; discuss the tensor decomposition algorithm considered] [PM: Discuss the results and compare][PM: Requested study on varying d(dimension) and #training (mesh plot)] [PM: Export this report in PDF and share with Dr. Puchades and me via email]