Classifying Character Trajectories with Echo State Networks

Pablo Prietz, 2108603

***Abstract*—In this project character trajectories of 20 different classes are classified using echo state networks for feature expansion and the k-nearest neighbourhood method for classification.**

I. Introduction

Imagine a resting water surface. If you start perturbing it a wave pattern will form. If the perturbation happens on multiple locations in a short amount of time the wave pattern will overlap and influence each other. When you stop perturbing the water it will come back to its resting state after some time. These *liquid states* contain spatial and temporal information about the perturbation. The longer a perturbation is ago the less information about this perturbation is carried by the liquid. It has fading memory. You can learn what the perturbation looked like depending on the wave pattern.

A. Reservoir Computing

Reservoir computing picks up this idea of liquid states. Instead of using water as liquid it uses a recurrent neural network which is only sparsely connected. The recurrent nature of the network makes sure that different perturbations can interact with each other. The sparse connectivity ensures that the signal does not propagate too fast and the temporal information is not lost. Each neuron in the network is a high-dimensional dynamical system. The idea is that the input (the perturbation) is projected into the high dimensional space of the neurons. A feature expansion takes place. A set of read out neurons (which can be part of the recurrent network) are able to transform the high dimensional state into a linear output stream. In this neural network the hidden layer, the recurrent part, is randomly connected and its weights are fixed. Only the weights of the output are trained. This computational model was proposed by Maass, Wolfgang and Natschläger in 2002[2]. It is a biological model and the signaling between the neurons is binary. Based same idea the echo state network (ESN) was developed independently by Jäger in 2001[1]. Instead of using a biological model for its neurons it uses the classical perceptron with a $tanh$ based activation function.

B. Project Goal

The goal of this project is to implement an ESN and test its classification power. The task which will be performed is the categorization of character trajectories. More information on the task can be found in section II.

P. Prietz is with the Department of Cognitive Science, University of Osnabrück, Osnabrück, Germany

II. Problem Definition and Algorithm

A. Data Set

I use the UCI character trajectories data set by Ben H Williams[3]. It is a low dimensional real-valued time-series with 20 classes. Important information about the data set:

> "The characters here were used for a PhD study on primitive extraction using HMM based models. The data consists of 2858 character samples [...]. The data was captured using a WACOM tablet. 3 Dimensions were kept - x, y, and pen tip force. The data has been numerically differentiated and Gaussian smoothed, with a sigma value of 2. Data was captured at 200Hz. The data was normalised [...]. Only characters with a single 'PEN-DOWN' segment were considered. Character segmentation was performed using a pen tip force cut-off point. The characters have also been shifted so that their velocity profiles best match the mean of the set."

Additionally all zero columns are removed. Since the data is very low dimensional there is a great potential for the feature expansion of the ESN. This also allows a small network size of 20 - 30 neurons. This increases the computational performance.

B. Procedure

The ESN has 25 neurons and a sparsity of 0.1. It is initialized with random weights in the range of $[-1, 1]$. The input weights are scaled with 0.4 and the recurrent weights with 0.3. This avoids self oscillating behavior of the network. The network is warmed up with a random time series with values in range of the given samples. The last state of the warm up is saved as reset point. This makes sure that each time series has the same starting point for training and testing.

In the next step the ESN is perturbed with the training and test time series. After each time series the network is reset to the mentioned reset point. The $last-i$ network states where $i \in \{140, 120, 100, 80, 60, 40, 20, 0\}$ were recorded. This results in a $(25 \cdot 8) \times n$ matrix where n is the total number of training and test samples. Each sample corresponds to one time series.

After the feature expansion of the ESN the resulting network states are classified using the 5-nearest neighbours method.

III. Experimental Evaluation

The data is partitioned into training and test sets using the 10-fold cross validation method. The ratio between correctly classified test samples to the total amount of test samples is used as correctness measure. Table I shows the performance using different amounts of stored network states. It shows that

Table I shows the performance of the classification process using different amounts of network states per time series. The performance is measured as ratio between correctly classified test samples to the total amount of test samples.

TABLE I. CLASSIFICATION PERFORMANCE

$last - i$ stored network states, $i \in$	Mean	Standard Deviation
$\{150, 140, 130, 120, 110, 100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 0\}$	0.940510	0.013933
$\{140, 120, 100, 80, 60, 40, 20, 0\}$	0.912179	0.020809
$\{120, 80, 40, 0\}$	0.851287	0.027645
$\{80, 0\}$	0.707503	0.037628
$\{0\}$	0.397478	0.022247

the performance increases with higher usage of time slices. It is highly probable that including too many network states will lead to overfitting.

The implementation and comparison to other categorization algorithms would be beyond the limits of this project and are sadly missing here. Also the validation of hypotheses using statistical tests is missing.

IV. CONCLUSION

Reservoir computing can lead to very good classification if the data meets the right conditions. For the feature expansion of the ESN a low dimensional time series with a lot of time steps would be ideal. These conditions are mostly given in this experiment. Which is one explanation for the good performance. Further comparisons to other data sets and classification algorithms would be necessary to make a meaningful statement about the classification performance of reservoir computing.

REFERENCES

[1] Herbert Jaeger. "The echo state approach to analysing and training recurrent neural networks-with an erratum note". In: *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148 (2001), p. 34.

[2] Wolfgang Maass, Thomas Natschläger, and Henry Markram. "Real-time computing without stable states: A new framework for neural computation based on perturbations". In: *Neural computation* 14.11 (2002), pp. 2531–2560.

[3] Ben H Williams. *Character Trajectories Data Set*. URL: http : / / archive . ics . uci . edu / ml / datasets / Character + Trajectories.