Classification of Human Motion using Echo State Neural Networks

Ernesto Rodriguez

Computer Science
Jacobs University Bremen
College Ring 7
28759
Bremen
Germany

Type: Guided Research Proposal
Date: December 9th, 2012
Supervisor: Prof. Dr. Herbert Jäger

Executive Summary

Echo state neural networks can be trained to simulate periodic time series such as human motion. Different neural networks can be trained to simulate different time series. Then given a time series of unknown origin, it can be classified by measuring which of the networks best models the unknown time series. Classification of time series is important for a wide variety of fields and different approaches are chosen depending on the time series being studied. Human motions stored as motion capture data are a time series in dire need for classification. The applications range from SLAM to motion indexing and search. This research consists on training Echo State Neural Networks to simulate various human motions and use them to classify unknown human motions encoded as motion capture data. The performance of this method will be compared to the performance of the more traditional Motion Templates approach.

---

1 Introduction

Motion capture is a method used to digitalize human motion. It is usually used to create realistic computer animations. The emergence of computer animated films and games made motion capture significantly important since it is the leading technology used to animate characters. The video game and film industry invest considerable amount of money to obtain motion capture data of new movements. According to [10], new movements could be reconstructed from existing motion capture data which would be cost effective, particularly to small companies.

The classification of motion capture data is of particular interest for searching & indexing motion capture data [4, 10], SLAM [2] and sign language recognition [6]. The film and video game industries work with huge amounts of motion capture data. Automatic motion capture data classification is needed to build motion capture databases with efficient querying mechanisms. This is a challenging task since its resolution resides in the classification of time series.

There exists several approaches to classify motion capture data. Motion Templates [7] classify the data by inspecting semantic properties of human movement. The classification is done by determining which properties are present and absent in a particular motion. Motion Templates have the weakness that the semantic properties must be manually specified which makes them inconvenient when dealing with several different motion types. Match Webs [4] is a searching method that finds motions that are numerically similar to the query and uses the results as new queries to obtain more results. This method can only search but not classify motion capture data and requires big databases to work. It can also have difficulties dealing with unaligned motions since it would require many queries to reach numerical similarity. Self Organizing Maps [10] is a learning approach which converts the motion capture data into a two dimensional map which is then clustered. It requires that the data is aligned in time which is done by pre-processing the data with the Smith-Watterman algorithm. Self Organizing Maps are very successful for classification and segmentation of motion capture data. There exist meth-

ods to classify time series which can be used for motion capture data. The Extended Frobbenius Norm (eros) [12] is a metric that uses eigenvector matrices of the SVD decomposition of the covariance matrices of two time series to measure the distance (or similarity) of the time series. This method can be used to classify motion streams by measuring the similarity among them. Since this is a statistical approach, it might ignore semantic properties of the motions such as the importance of the hand grip while differentiating grabbing and touching an object.

Echo state neural networks have been very successful to learn and simulate time series [3]. Since human motion is a time series, echo state neural networks should be able to learn various human motions effectively. This research will exploit the learning power of echo state neural networks to create a new motion capture data classifier. This method will be capable of extending its scope by learning new motions such that they can be further classified. The method will be compared to other existing methods and further analyzed to determine if it is suitable for automatic segmentation of motion capture data.

2 Technologies Employed

2.1 Echo State Neural Networks

Neural networks are computational mechanisms inspired from the way brains work. They consist of input neurons, neurons (called internal or hidden units) and output neurons. Depending on the way the neurons are connected to each other they are classified as recurrent neural networks or feed-forward neural networks. Echo state neural networks (ESNs) are a type of recurrent neural networks whose primary characteristic is that only connections from neurons to outputs are trained. Echo state neural networks are a powerful mechanism to simulate time series as presented in [3]. The remainder of this section will be used to formally present echo state neural networks as they will be used in the research, which is based on [3].

A time discrete network will be considered. The network has N internal network units and L output units. Note that networks without input units will be used in this research. The state of the network at a given time t is $\mathbf{x}(t) = (x_1(t), x_2(t), .., x_N(t))$ and the output of the network at time t is $\mathbf{y}(t) = (y_1(t), y_2(t), .., y_L(t))$. The connection weights between units are collected in matrices. A $N \times N$ matrix $\mathbf{W}$ for the internal unites, a $L \times (N + L)$ matrix $\mathbf{W^{out}}$ for connections between internal units and output units and a $N \times L$ matrix $\mathbf{W^{back}}$ output feedback matrix. The state, weights and outputs will all be real valued numbers.

The state of the network is updated according to the equation

$$\mathbf{x}(t + 1) = \mathbf{f}(\mathbf{W}\mathbf{x}(t) + \mathbf{W^{back}}\mathbf{y}(t))$$

where,

- $\mathbf{f} = (f_1, f_2, ..., f_N)$ are the output functions of the internal units (usually sigmond functions).

The output of the network at time t is given by the equation

$$\mathbf{y}(t+1) = \mathbf{f^{out}}(\mathbf{W^{out}}(\mathbf{x}(t), \mathbf{y}(t)))$$

where,

- $\mathbf{f^{out}} = (f_1^{out}, f_2^{out}, ..., f_L^{out})$ is the output units functions
- $(\mathbf{x}(t), \mathbf{y}(t))$ is the concatenation of the internal state and output vectors

The training procedure for ESNs goes as follows. Suppose we are given the output signal $\mathbf{y_{teach}}(t)$ for $t = 0, .., t_{max}$. We wish to have a ESN whose output signal $\mathbf{y}(t)$ is as close as possible to $\mathbf{y_{teach}}(t)$. This is done as follows.

Produce a ESN with N input units and L outputs where L is the dimension of the vector $\mathbf{y}(t)$. Random sparse matrices are used to produce the $\mathbf{W}$ and $\mathbf{W^{out}}$. The network must have echo states which is usually the case if $|\lambda_{max}| < 1$ where $|\lambda_{max}|$ is the eigenvalue of $\mathbf{W}$ with the greatest absolute value. A detailed description on how to produce such network can be found in [3].

Run the network by teacher forcing the $\mathbf{y_{teach}}(t)$ signal into the network. Teacher forcing consists of replacing the network output $\mathbf{y}(t)$ with the value of $\mathbf{y_{teach}}(t)$. This means that the update function of the network is:

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{W}\mathbf{x}(t) + \mathbf{W^{back}}\mathbf{y_{teach}}(t))$$

The network must be run in this way for some initial transient and the outputs during this transient will be discarded. It is better to discard a big amount of initial states, but the amount of data discarded also depends on the amount of training data available.

Collect the ESN output for the remaining training data. We will denote with t_{min} the time where the output began to be collected and t_{max} the time where the last output was collected. Now to complete the training, our task is to minimize the following error function:

$$mse_{train,j} = 1/(t_{max} - t_{min}) \sum_{t=t_{min}}^{t_{max}} (g'_j(t) - \sum_{i=1}^{N} w_{j,i}^{out} x_i(n))^2$$

where,

- $g'_j(t) = ((f_j^{out})^{-1})(y_{teach,j}(t))$ for $j = 1, .., L$ is the inverse output function applied to the jth component of the teacher signal.

New entries for the $\mathbf{W^{out}}$ matrix must be calculated such that they minimize the error function above for all components. One particular method of doing this is by solving the equation:

$$\mathbf{X W_j^{out}} = \mathbf{c_j}$$

where,

- $\mathbf{X} = (\mathbf{x}(t_{min}), .., \mathbf{x}(t_{max}))^\intercal$ is a matrix where the ith row contains the ESN state $\mathbf{x}(t)$ at time $t = i$, $i = t_{min}, .., t_{max}$. In other words the rows of the matrix contain all states that the network encountered during training
- $\mathbf{c_j} = (\mathbf{y_{teach}}(t_{min})_j, .., \mathbf{y_{teach}}(t_{max})_j)^\intercal$ is a vector which contains in it's ith component the value of the jth component of the training function $\mathbf{y_{teach}}(t)$ at time $t = i$, $i = t_{min}, .., t_{max}$.
- $\mathbf{W_j^{out}}$ is the jth column of the new $\mathbf{W^{out}}$ matrix.

This can be solved using the Moore-Penrose psudoinverse [11] with the following equation:

$$\mathbf{W_j^{out}} = \mathbf{X}^+ \mathbf{c_j}$$

where,

- $\mathbf{X}^+$ is the Moore-Penerose psudoinverse of the matrix $\mathbf{X}$

Please refer to [3] for more information about training ESNs.

2.2 Motion Capture Data

Human motion is captured by placing identifiers on a human's joints and tracking the motion of these identifiers with cameras. The obtained data is raw x,y,z coordinates with the position of the joints at a particular time. This data is further processed and encoded into different motion capture data formats. The measuring errors from cameras are reduced by fixing the lengths of each joint and the data is then encoded in any of the various motion capture data formats [5]. Depending on the needs, different formats have advantages against other formats, but in all cases the human motions are represented using human skeletons and angles.

The motion capture data from HDM05 database [1] is available in Coordinate 3D (C3D) and Acclaim Motion Capture (AMC) formats. The C3D format is binary and the ACM format is plain text. Therefore the ACM format will be preferred in this research since is easier to extract the data from. The AMC files in this library consist of 29 joints with a total of 62 degrees of freedom. The data is encoded in frames and each frame specifies the value of the 62 entries.

The position in space of each of the joints can be calculated by starting from the root joint and adding the length of the joint after applying the rotational transformations specified for that joint. This process is then repeated to reach the next joints. It is possible that many joints extend from a single joint. There exist software to parse these file formats, such as a importer for Matlab available in [1].

3 Objectives and Evaluation

The objectives of this research are the following:

- Develop a motion capture data classifier based on echo state neural networks
- Develop a method to reduce the dimensionality of motion capture data and still archive a successful classification
- Benchmark the method with the data from HDM05 [1]
- Compare this method with existing methods, particularly with Motion Templates [7] and Self Organizing Maps [10]
- Evaluate the possibility of extending the method with automated segmentation

The first aspect to be evaluated is the reduction of dimensionality that will be attempted. It is expected that a classifier can still be built even after reducing the dimensions of the motion capture data. If possible, the classifier will be tested against the performance of a slower classifier which uses the data without reduction.

This method will be directly compared with Motion Templates [7] since the HDM05 database was created for that research. This will provide a general overview on how ESNs perform against traditional methods to classify motion capture data but will not provide information about how it performs against state of the art approaches like SOM [10]. The reason is that SOMs go a step further and their performance is measured by segmenting continuous motion streams. This would be the next step for this method if it proves to be effective.

Finally, it is important to determine if ESNs can handle movements that are not aligned, especially in time. In [10], the Smith-Waterman algorithm is used to align the motion capture data before the classification. This research will evaluate whether time warping is an important factor by testing the classifiers with aligned and unaligned data.

4 Planned Investigation

The data that will be used in this research is the data from the HDM05 motion capture database. This database contains over 3 hours of motion capture data. There are 70 different types of human movements available and each was preformed between 10 and 50 times by various actors. Not all of the movements will be considered in this research, but rather the ones with plenty of available data to ensure there is enough training and testing data. All the data is encoded as specified in section 2.

The investigation will begin by building a ESN that can simulate human motion encoded as motion capture data. It was pointed out in [10] that using the angles of the skeleton in motion capture data eliminates the difficulties of having different sized bones. Thus the first naive approach will be using a ESN with 59 outputs since the absolute position of the skeleton will be discarded. This should prove to be a very accurate simulation but computationally expensive. The number of outputs will then be reduced to archive better performance. The following steps will be taken:

- Removal of unnecessary bones
- Extraction of features

The removal of unnecessary bones can be done up to some degree by intuition. Further removal will require a more advanced analysis. One possibility is calculating the conditional entropy [8] to determine which angles present high entropy given the rest of the angles for all movements considered.

To determine whether the networks obtained are suitable, they will be trained and run by teacher forcing test movements. The error obtained by comparing the output of the network and the movement which was teacher forced into the network should be on average higher if the network was not trained for that movement. Formally, given a set of movement types $M = \{m_1, m_2, .., m_n\}$ we have a set of samples for each movement $S = \{s_{m_1}, s_{m_2}, .., s_{m_n}\}$. Each s_{m_i} sample will be divided in $train_{m_i}$ and $test_{m_i}$. A set of ESNs will be produced $E = \{e_{m_1}, e_{m_2}, .., e_{m_n}\}$ and each e_{m_i} will be trained with the movements of $train_{m_i}$. Then e_{m_i} is driven by teacher forcing the movement $\mathbf{y_{test}}(t) \in test_{m_j}$.

The following error can be defined for each $m_i \in M$ for the network e_{m_j}:

$$mse_{e_{m_j},m_i} = \frac{1}{|test_{m_i}|} \sum_{\mathbf{y_{test}} \in test_{m_i}} \frac{1}{n} \sum_{k=0}^{n} (\mathbf{y_{test}}(k) - \mathbf{y_j}(k))^2$$

where,

- $\mathbf{y_j}(t)$ is the output of the ESN e_{m_j} at time t driven by teacher forcing the motion $\mathbf{y_{test}}(t-1)$
- n is the length of the movement $\mathbf{y_{test}}$.

Given this error, $mse_{e_{m_i},m_i} < mse_{e_{m_i},m_j}$ for all $i, j < n$, $j \neq i$. If the conditions are favorable the difference will be significant for all cases.

After building the set of expert ESNs E, the classification performance of the networks will be evaluated. A collection with samples of every movement S_{eval} will be built. A reference set $R := \{((\mathbf{y_i}, m_i) | \mathbf{y_i} \in S_{eval} \wedge \mathbf{y_i} \in s_{m_i}\})$ will contain pairs matching a movement and the type of that movement. All these movements will be used to test the performance of the classifier. All of the movements $\mathbf{y_{test}}(t) \in S_{eval}$ will be used to drive each of the ESNs in E and the output of the ESN will be compared with the values of $\mathbf{y_{test}}(t)$ for all times t using the following error function:

$$mse_i^{classify}(\mathbf{y_{test}}) := \frac{1}{n} \sum_{j=0}^{n} (\mathbf{y_{test}}(j) - \mathbf{y_i}(j))^2$$

where,

- $\mathbf{y_i}(t)$ is the output of the ESN e_{m_i} driven by teacher forcing the the movement $\mathbf{y_{test}}(t-1)$.

After calculating the $mse_i^{classify}$ of a particular movement for all e_{m_i} networks. The smallest $mse_i^{classify}$ error will be selected and the given motion $\mathbf{y_{test}}$ will be classified as m_i which will be represented by the result pair $r_{test} := (\mathbf{y_{test}}, m_i)$. It will be considered a successful classification if $r_{test} \in R$ or a misclassifications otherwise. Finally, the performance of the classifier will be calculated by dividing the number of successful classifications by the total number of samples given.

5 Preliminary Work

An implementation of ESNs based on [3] is being developed for this research and the code is available in [9]. The code includes functionality to generate ESNs of variable sizes, train the ESNs using the Moore-Penrose psudoinverse and run the ESNs either by teacher forcing or regular means. This implementation has been evaluated for simple series such as sine waves which resulted in good approximations. There is still work to be done since chaotic time series such as the Henon time series do not perform well with this implementation. The final classifier will be based on this implementation which is expected to become more powerful.

6 Timeline

- February 15th, 2013: Complete the development of the code to create, train and run the ESNs and the code used to extract the motion capture data.
- March 15th, 2013: Complete the creation of the expert neural networks.
- April 15th, 2013: Complete testing the classifiers. Measure the performance with aligned and unaligned motion capture data.
- May 15th, 2013: Final report submission.

References

[1] Motion capture database HDM05. `http://www.mpi-inf.mpg.de/resources/HDM05`

[2] Slawomir Grzonka, Andreas Karwath, Frederic Dijoux, and Wolfram Burgard. Activity-based estimation of humman trajectories. *IEEE Transactions on Robotics*, 1(28):234–245, 2012.

[3] Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks - with an erratum note. Technical report, Fraunhofer Institute for Autonomous Intelligent Systems, 2010.

[4] Lucas Kovar and Michael Gleicher. Automated extraction and parametrization of motions in large data sets. *Transactions on Graphics*, 23(3), 2004.

[5] Jeff Lander. Working with motion capture data file formats. Technical report, Game Developer, 1998.

[6] Chuanjun Li, S. Q. Zheng, and B. Prabhakaran. Segmentation and recognition of motion streams by similarity search. *ACM Transactions on Multimedia Computing, Communications and Applications*, 3(16), 2007.

[7] Meinard Müller and Tido Röder. Motion templates for automatic classification and retrieval of motion capture data. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 137–146, 2006.

[8] Kaustubh Pathak. Artificial Intelligence. Lecture slides for the artificial intelligence course at Jacobs University, 2011.

[9] Ernesto Rodriguez. Lambda neural networks. `https://github.com/netogallo/LambdaNN`.

[10] Wu Shuangyuan, Xia Shihong, Zhaoqi Wang, and Chunpeng Li. Efficient motion data indexing and retrieval with local similarity measure of motion strings. *Visual Computer*, 25:499–508, 2009.

[11] Eric W. Weisstein. Moore-penrose matrix inverse. From MathWorld – A Wolfram Web Resource. `http://mathworld.wolfram.com/Pseudoinverse.html`.

[12] Kiyoung Yang and Cyrus Shahabi. A pca-based similarity measure for multivariate time series. In *Proceedings of the 2nd ACM International Workshop on Multimedia Databases*, pages 65–74, 2004.