A Neural Classifier for Anomaly Detection in Magnetic Motion Capture

Iain Miller[1] and Stephen McGlinchey[2]

[1] University of Paisley, Paisley. PA1 2BE, UK
iain.miller@paisley.ac.uk,
[2] stephen.mcglinchey@paisley.ac.uk

Abstract. Over recent years, the fall in cost, and increased availability of motion capture equipment has led to an increase in non-specialist companies being able to use motion capture data to guide animation sequences for computer games and other applications.[1] A bottleneck in the animation production process is in the clean-up of capture sessions to remove and/or correct anomalous (unusable) frames and noise. In this paper an investigation is carried out into whether the 2-layer SOM network previously designed [5] and trained on one capture session, can be used to create a neural classifier to be used to classify another separate capture session.

1 Introduction

Motion capture is the process of recording the motion of actors and/or objects, and this data is often used in computer games to animate characters and other game objects. The process normally involves tracking sensors or markers that have been placed in key positions on the actor's body, and detecting their locations in three-dimensional space. As the cost of equipment decreases, the realm of Motion Capture is no longer the preserve of specialist companies who take care of all aspects of data capture and post-processing. The task of supplying animation scenes from a motion capture system is now seen as a commodity, and so the focus has started to veer towards processing the output from a capture session as quickly and cheaply as possible. By improving post-processing, motion capture studios can get more useful (and commercial) application out of the capture equipment.

Previous work [4] showed that a statistical method could correctly identify a frame's class 67% of the time, whilst [5] showed that a 2-layer SOM network could produce an output net that grouped classifications together. From this base, the remit of this paper is to investigate the extent to which a network similar to that identified in [5] trained on one set of data could be used to classify data from a different capture session. Other research, such as that in [2], [3] and [6] focuses on the extraction of information from data that is already clean. Currently, it is part of an animators job to clean the raw data from a capture session, and it is this time that we wish to save for the animator.

Positional anomalies in the data come about when the sensors move too close to or too far from the field generators; where the sensors are unable to detect the field strength accurately or if metallic objects interfere with the magnetic fields and so produce anomalous results. The outcome being sensors reporting their positions that are inverted in the vertical axis or placed at a seemingly random position and breaking the skeleton of the captured article (which can be human, animal or an inanimate object). The next section outlines the structure of the network and the factors involved in designing the classifier.

2 Methodology

Data is read in and stored in a separate matrix for each sensor in the session (called nodes from here on). Equation 1 shows one node's data in one frame in the session.

$$n_i(t) = \left[n_{i1}(t)\ n_{i2}(t)\ n_{i3}(t) \right] \tag{1}$$

For each node, a one-dimensional SOM is created and initialised (equation 2 with M as the number of neurons in the net). Each SOM is trained for 100 epochs using only the data for its associated node. One epoch uses every frame in the session, fed into the network in a random order.

$$S_i = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_M \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \\ & \vdots & \\ s_{M1} & s_{M2} & s_{M3} \end{bmatrix} \tag{2}$$

The Euclidean distance between the input vector and each neuron in a SOM is calculated, and the neuron with the minimum distance being declared the winner (see equation 3, i is the node number and k is the vector element).

$$c_i = \arg \min_{1 \le j \le M} \left(\sqrt{\sum_{k=1}^{3} (n_{i_k}(t) - s_{j_k})^2} \right) \tag{3}$$

The weights for the winning neuron are then updated using equation $s'_l = s_l + \alpha h(n_i(t) - s_l)$ with α being the adaptive learning rate ($\alpha = \alpha_0(1 - \frac{tc}{T})$, $T = 100F$, where F is the total number of frames and tc is the training cycle), and $h_1 = e^{\frac{-(j-c_i)^2}{2}}$ is the gaussian neighbourhood function (j and c_i are the indices of the neuron being updated and winning neuron respectively) that modifies the neurons closest to the winner more than those further away.

The outputs from each of these SOMs form the input vector for the second-layer SOM (equation 4). The second-layer SOM is a two-dimensional array of neurons with each neuron having a weight vector of that shown in equation 5. The winner is decided by the minimum Euclidean distance, as in the first-layer SOMs, with the training updates calculated using the same adaptive learning rate and gaussian neighbourhood function $h_2 = e^{\frac{-(R-c_R)^2-(C-c_C)^2}{2}}$ (with R and

C being the row and column address of the neuron being updated and c_R and c_C the row and column address of the winning neuron).

$$In = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_M \end{bmatrix} \tag{4}$$

$$v = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} \tag{5}$$

2.1 Centring Data

The above method does leaves the positional data as it is and does not apply any pre-processing onto it. By leaving the data unaltered, the network can take account for the spatial element of the capture volume. However, when using a test-case to train a neural-classifier, it may be appropriate to remove this spatial element of the data, in order to concentrate the network to train on just the structural element of the capture. To do this, the positional data should be translated to the origin in each frame.

There are several methods of being able to centre the data, e.g. centre of mass, etc. Here, the centre of mass version is used. With this method each node in a frame is given an equal mass and then the geometric centroid of the masses is found using equation 6, but as the masses are all equal this can be simplified to equation 7.

$$\bar{n}(t) = \frac{\sum_{i=1}^{M} m_i n_i(t)'}{\sum_{i=1}^{M} m_i} \tag{6}$$

$$\bar{n}(t) = \begin{bmatrix} \bar{n}_x(t) \\ \bar{n}_y(t) \\ \bar{n}_z(t) \end{bmatrix} = \frac{\sum_{i=1}^{M} n_i(t)'}{M} \tag{7}$$

With $\bar{n}(t)$ giving the coordinates of the centroid. Each node in the captured frame is then translated to the origin in the x- and z- axes towards the origin of the volume space, equation 8.

$$g_i(t) = n_i(t)' - \begin{bmatrix} \bar{n}_x(t) \\ 0 \\ \bar{n}_z(t) \end{bmatrix} \tag{8}$$

The centred data is then used to train the network in exactly the same way as shown in equations 1 to 5, except with $g_i(t)$ replacing $n_i(t)$ in all relevant equations.

2.2 Classifier

From previous experiments [5] a network with 41 neurons in each first-layer SOM and a 21-by-21 SOM for the second-layer gave the most consistent results. Therefore for these investigations a network with this same structure was trained and used for classification.

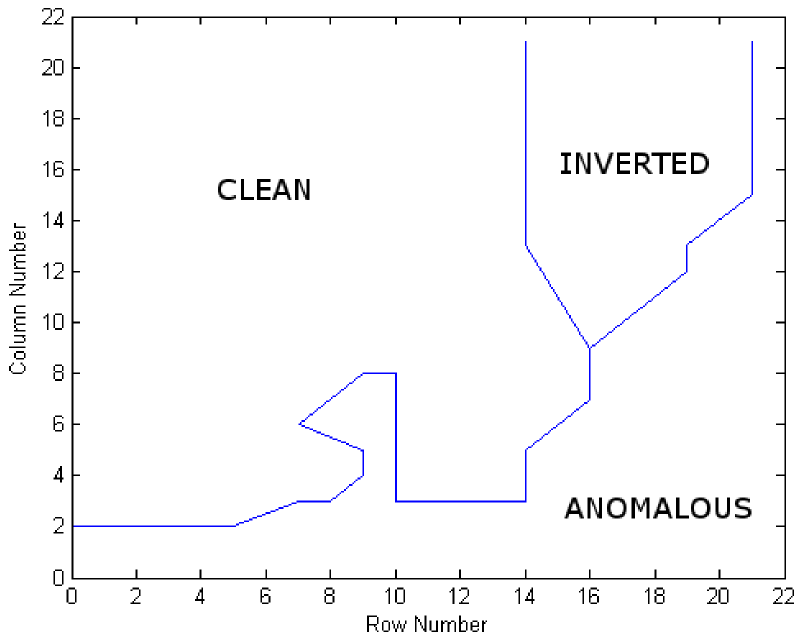

Fig. 1. A Graph Showing the Classifier Boundary for a 2-Layer SOM Network trained with Centred Data

The original outputs of the networks were used to create output classifier graphs (see figure 1 for an example) and filter. The class boundaries were defined by a user and the criteria used to decide where to draw the boundary were:

1. The boundary has to go through each Overlapping Point.
2. The boundary should go nearer the ”clean” neurons than the ”anomalous” neurons.
3. In areas of uncertainty, then take the nearest known neurons as reference.
4. The boundary represents the unknown areas

3 Results

The classifying networks were trained using a capture session of a single actor walking around (called Train1 from hereon), that contains two sequences of anomalous frames sandwiching a series of clean frames, all preceded by a series of frames where the figure is inverted. The networks were then used to classify 5 other files each with different characteristics. File F1 has an identical sensor set up to Train1 with very similar motions and orientations but is less than half the length. File T1 has two series of clean frames alternated with two series of anomalous data, file T2 has a series of clean frames followed by a series of anomalous frames with more clean than anomalous. File T3 is the same as T2 but with more anomalous than clean and File T4 contains only clean frames. Files T1, T2, and T3 all have 2823 frames with similar but not identical sensor setups to Train1 capturing a range of more energetic motions than in Train1. T4 also has 2823 frames, but has similar motions to Train1.

A false positive is where a anomalous or inverted frame is classified as a clean frame, with a false negative if a clean or inverted frame called as a anomalous and a false inverted is when a frame is called as inverted when it should be either clean or anomalous.

As can be seen from tables 1 and 2, for those files with similar motions to those captured in Train1 (files F1 and T4), that the results give us no or very few false readings. However, for file T4 the majority of frames are classified as unknown. On inspection of which neurons are producing these spurious results with the uncentred data 4 neurons have been mislabeled as unknown, when they should have been labeled as clean. With centred data, one neuron has been mislabeled as anomalous when it should have been labeled as unknown and one neuron has

Table 1. Table of results for classification using a 2-layer SOM trained with uncentred data

File	Correct			False			Unknown
	Positive	Negative	Unknown	Positive	Negative	Unknown	
Train1	115	213	46	0	0	0	33
F1	115	37	0	0	0	0	18
T1	299	876	0	393	798	32	425
T2	811	23	0	62	221	0	1706
T3	474	0	0	0	773	0	1576
T4	118	0	0	0	0	0	2705

Table 2. Table of results for classification using a 2-layer SOM trained with centred data

File	Correct			False			Unknown
	Positive	Negative	Unknown	Positive	Negative	Unknown	
Train1	131	225	49	0	0	0	2
F1	129	39	0	2	0	0	0
T1	381	584	0	1531	107	0	220
T2	2287	33	0	71	223	2	207
T3	1261	73	0	1379	13	3	94
T4	1074	0	0	0	4	0	1745

been labeled as unknown when it should have been labeled as clean. The files where the motions captured are significantly different from Train1 (T1–3) yield poor results, which suggests that it is only possible to use a neural classifier like this on files of similar motions to the trainer file.

4 Conclusions

Although it was initially hoped that a file that contained all possible classifications, such as Train1, would be able to create a classifier that could be used on a majority of other files with similar sensor set-ups. This isn't the case but it does show that it is possible to build a classifier that can correctly identify the majority of frames in a separate session with similar set-ups and motions. As was shown, the initial classifier requires a degree of guesswork from the user and so is not at its optimum level, but as further files are tested against the classifier it can be refined further improving performance. There will always be points in a classifier that will produce unknown classifications, due to the existence of overlapping neurons. From the two sets of results it appears on first inspection that centring the data has little effect, but it actually produced an initial classifier that required less refinements than the uncentred data. Part of this will be down to user-guestimates in building the classifier and part due to the centred data-trained network creating a clearer separate in groups of neurons.

To further advance the theory more testing needs to be done with a variety of captured motions, lengths of file and sizes of actors. Furthermore, a trainer file should be created that contains a full range of motions and orientations in itself, rather than in individual files.

References

1. Geroch, M.: Motion Capture for the Rest of us. Journal of Computing Sciences in Colleges. **19.3** (2004) 157–164
2. Gibson, D., Campbell, N., Dalton, C., Thomas, B.: Extraction of Motion Data from Image Sequences to Assist Animators. Proceedings of the British Machine Vision Conference (2000)
3. Kovar, L., Gleicher, M.: Automated Extraction and Parameterization of Motions in Large Data Sets. ACM Transactions on Graphics, **23.3** (2004) 559–568
4. Miller, I., McGlinchey, S.: Automating the Clean-up Process of Magnetic Motion Capture Systems. Proceedings of the Game Design and Technology Workshop (2005)
5. Miller, I., McGlinchey, S., Chaperot, B.: Anomaly Detection in Magnetic Motion Capture using a 2-Layer SOM network. To Appear in Proceedings of IEEE Conference of Computational Intelligence in Games 2006. (May 2006)
6. Müller, M., Röder, T., Clausen, M.: Efficient Content-Based Retrieval of Motion Capture Data. ACM Transactions on Graphics **24.3** (2005) 677–685