STUDYING AND IMPROVING LINEAR MAPPINGS BY ARTIFICIAL NEURAL NETWORKS

Emőd Kovács (EKTF, Hungary)

Abstract: The aim of this paper is to evaluate the effectiveness of artificial neural networks studying linear mappings and, on the other hand,improve deformed linear mappings given by wrong pairs of points. In this latter case the artificial neural network is applied to give the best fitting linear mapping of the given set of data.

1. Introduction

Artificial neural networks or simply neural nets are widely used in computer graphics e.g. in surface reconstruction from insufficient or scattered data [4]–[7]. Generally neural nets is a useful tool for handling any kind of data which have deformity or deficiency from a certain point of view. The main feature of the neural nets is the ability of studying, which means that the given data can improve the structure of the net. Neural nets can be classified by the type of input data or the method of studying [2]. In this paper the well-known back-propagation algorithm is used, which will be discussed in Section 2.

Our purpose was to use neural nets studying linear mappings from exact and also from deformed data. Planar linear mappings, like rotation or affine transformation play essential role in computer graphics. Even if we want to display spatial objects or movements with our computer, a classical parallel projection or other kind of (degenerate) linear mapping has to be used. Of course each of these linear mappings has a linear system of equations (or a matrix) which transforms the co-ordinates of the points. Hence in the first part of the research, when the neural nets have to be trained by these well-known mappings, we could evaluate the speed of training. Since every linear mapping has a crucial number of points, with which the transformation is uniquely determined, the nets are trained by that amount of pair of points (e.g. three pairs of points for a planar affine transformation).

However it can be happened, that among the data there are one or more 'false' pair of points. If five or more general pairs of points are given in the plane there is no exact linear transformation which maps the points onto their image points, since the most general linear transformation, the projective transformation is given by four pairs of points. Hence we can compute a system of equations from four pairs, but there is no guarantee that this mapping will transform the rest of the points onto their given images, usually the computed images and the given image points will be far from each other. With the help of the neural nets we will give a mapping for any number of given points, which will be linear and has the smallest error in the image points.

2. The neural network and the back-propagation algorithm

Studying the planar linear mappings the applied neural network is a two layered network with two or three input nodes in the first layer and two or three output nodes in the second layer entirely connected to each other. The layers consist of two or three nodes according ot the current transformations e.g. we use projective co-ordinates to describe the projective mappings, and points has three projective co-ordinates in the plane (for the use of projective geometry see [1]). Since neural nets and especially the back-propagation algorithm are well-known computational tools, we give only a short description of the algorithm referring mainly the differences between the widely used method and the present one. For a more detailed survey see e.g.[3].

The main difference, as we can see from the algorithm presented below, that since we want to compute linear mappings, the nodes of the neural net has no the generally used sigmoid function to compute the output, but a simple weighted sum is computed instead. Hence some of the training rules which would consist the derivative of the sigmoid function, will be simplified.

STEP 1. Set all weights w_{ij} to small random values (where w_{ij} denotes the weight associated to the connection between the i^{th} node of the input layer and the j^{th} node of the output layer.

STEP 2. Present a randomly chosen input, i.e. the co-ordinates of an input point.

STEP 3. Calculate the output, i.e. the weighted sum of the coordinates

$$\text{output}_j = \sum_i w_{ij}\,\text{input}_i$$

STEP 4. Adapt weights by the equation

$$w_{ij} = w_{ij} + \eta\delta_j\,\text{input}_i$$

where $\eta \in [0,1]$ is the so called gain term, while δ_j is the difference between the desired and the received output value in the j^{th} output node.

STEP 5. Repeat by going STEP 2. until the net is trained.

The network is said to be trained if all the outputs fit the desired output points, or, in case of over-defined data, if the changes of the weights fall under a predefined limit.

3. Training by well-defined linear mappings

Basic theorems of geometry state how many pairs of points determine uniquely a certain transformation in the plane. Hence if we want to define a rotation, an affine or a projective transformation, we have to give two, three or four pairs of general points (any three among them should be non-collinear).

Our question is to evaluate how fast the neural network can be trained by a set of data described above. On the other hand, we want to examine how exact the training is, since after the training procedure theoretically the weights should be equal to the coefficients of the appropriate system of equations. Indeed, if a general affine transformation is given by the equations

$$\tilde{x} = a_{11}x + a_{12}y + a_{13}$$
$$\tilde{y} = a_{21}x + a_{22}y + a_{23}$$

then comparing it with the equations yield the output of the network

$$\text{output}_1 = w_{11}\,\text{input}_1 + w_{21}\,\text{input}_2 + w_{31}\,\text{input}_3$$
$$\text{output}_2 = w_{12}\,\text{input}_1 + w_{22}\,\text{input}_2 + w_{32}\,\text{input}_3$$

where $\text{input}_3 = 1$, one can easily see, that the $a_{ji} = w_{ij}$ must hold for all i, j.

In case of well-defined linear mappings every run was successful under 1100 iteration (accuracy was 10^{-5}) and the training was exact. The following table shows the results after 1000 runs.

transformation	iterations
translation	343
rotation	400
affine tr.	500
projective tr.	1068

Our other table shows how the number of iteration changes in term of accuracy, in case of projective transformation.

accuracy	iterations
10^{-5}	500
10^{-10}	916
10^{-15}	1705

4. Training by over-defined data

If we consider an arbitrary linear mapping, then a certain number belongs to that mapping, which shows how many pairs of points define the mapping uniquely. If the number of input points exceeds this limit then our data set is said to be over-defined.

An over-defined set of data does not necessary mean false data set. We can compute the image of several points by a transformation, and if these pair of points are considered as the input data, the neural network has to produce exactly the same transformation. If we consider an affine transformation and 50 points and their images as input data, the training of the net will be slower than in the case of three points, but the same transformation will be received.

This table shows how the number of iterations changes if the number of points increases. Since one iteration means to feed all of the given points as input for the net, the total number of input simply the product of the number of iterations and the number of input points.

points	iterations
3	500
10	61
100	6

However over-defined data set could mean arbitrary pairs of points, the number of which is greater than the necessary number of data. In this case the set could be called false data, since there is no transformation which could map these points to their images. More precisely, if we consider affine transformations and the number of input points is 4, then there is no affine transformation for these points (of course a projective transformation can be easily computed from these data). But if we have 5 or more pairs of arbitrary points, then generally there is no any kind of linear transformation mapping these points to their images. This problem can occur e.g. as a wrong scanning or digitalisation of an image. Of course the false data are normally not completely wrong, perhaps only a corner of a figure will be curved, but the theoretical problem remains the same.

In this very case one can try to find a mapping which produced this image, but that transformation will not be linear, or can find a linear mapping which is close to the original one, that is the input points will be mapped almost to their images. Neural networks are applicable for both problems, but in this paper we consider only the latter case.

If we have a set of false data, first we have to decide which type of linear mapping is the desired. The main difference is, that affine transformation will preserve parallelism, but probably the difference between the desired and actual output will be larger. If we need smaller error, projective transformation has to be chosen.

Since the neural net does not 'know' if the set of data is correct or false, the training algorithm will be the same described above, however the number of iterations can be increased significantly. The error of the transformation can be measured by corrupting original transformations. Since neural networks minimize the cost function equal to the mean square difference (see [2]), the error (measured by Euclidean distance of the desired and actual output) will be lower, than the squareroot of the distortion of the original transformation.

5. Conclusion

In this paper artificial neural net with back-propagation training algorithm will be used to study linear mappings. First the effectiveness of the net will be discussed when exact linear transformation will be trained from a set of input data, the number of which were the same as the theoretical limit for the unique determination of the transformation. The training was succesfull and fast, even for larger number of input points.

On the other hand correction of corrupted linear transformation has been discussed. If the number of input points are exceeds the limit mentioned above, then there is no theoretical way to find linear mapping which transforms the given data to their images. Here arbitrary set of input points was given, and the neural net, trained by these data, has found the best fitting linear mapping.

References

[1] Herman, I., The Use of Projective Geometry in Computer Graphics, *Lecture Notes in Computer Science* 564, Springer-Verlag, 1991.

[2] Lippmann, R. P., An Introduction to Computing with Neural Nets, *IEEE ASSP Magazine*, April 1987, 4–22.

[3] Rojas, R., Neural Networks. A Systematic Introduction, Springer-Verlag, 1996.

[4] Hoffmann M., Várady L., Free-form curve design by neural networks, *Acta Acad. Paed. Agriensis*, Vol. XXIV., 1997, 99–104.

[5] Hoffmann, M., Várady, L., Free-form Surfaces for Scattered Data by Neural Networks, *Journal for Geometry and Graphics*, Vol. 2, No.1, 1998, 1–6.

[6] Várady, L., Hoffmann, M., Kovács, E., Improved Free-form Modelling of Scattered Data by Dynamic Neural Networks, *Journal for Geometry and Graphics*, Vol. 3, No.2, 1999, 177–181.

[7] Hoffmann, M., Modified Kohonen Neural Network for Surface Reconstruction, *Publ. Math. Debrecen*, Vol. 54 Suppl., 1999, 857–864.

Emőd Kovács
Institute of Mathematics and Informatics
Károly Eszterházy Teachers' Training College
Leányka str. 4–6.
H-3300 Eger, Hungary