Least Squares Finite Element Interpolations for Analysis of Model Free Problems

Pedro V. Marcal,
MPACT Corp.
pedrovmarcal@gmail.com

Jeffrey T. Fong
National Institute of Standards & Technology, Gaithersburg, MD 20899, U.S.A.

Robert Rainsberger
XYZ Scientific Applications, Inc.,
2255 Morello Avenue, Suite 220
Pleasant Hill, CA 94523
rains@truegrid.com

Abstract
A least squares Finite Element interpolation procedure was developed to interpolate a large data set of handwritten numbers. A full quadratic Lagrange Polynomial function was used for a finite element grid. The solution was developed for the MNIST data with a training set of 50,000 greyscale images consisting of 784 pixels each. The solution was local and required minimal computing resources. The process was compared to one which used Deep Learning, which was global and required large computing resources.

Introduction

The Finite Element (FE) technology [1] is based on local interpolation functions. It has been widely applied to solve engineering problems with great accuracy. In this context the problems may be interpreted as interpolations of geometry subject to a set of partial differential equations of equilibrium, e.g. [2]. Though it has not been applied in AI to resolve model free problems, there is no reason why it could not be effective in that arena in a simpler descriptive geometry problem. It has some natural advantages in that a mesh can be generated to represent any physical shape and or dimension. The degree of polynomial adopted for the FE also determines the order of the accuracy that can be expected. The quadratic form of an element results in a 9 node quad and a 27 node hexa element in two and three dimensions respectively. Nielsen [3] has studied the problem of recognizing hand written

numbers using deep learning. The data required for the problem is given by [4] and is referred to as the MNIST data. It consists of 50,000 images for training-data, another 10,000 each for validation-data and test-data, respectively. An image is represented as a 28 X 28 pixel grey scale, (0.0 to 1.0). A typical sample is given below in Fig. 1.

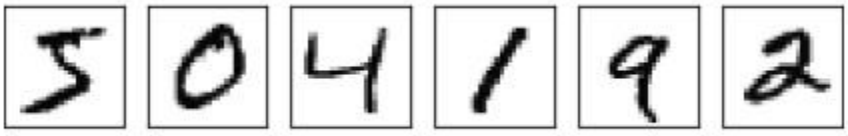

Fig. 1 MNIST sample images, six images of 28 X 28 greyscale pixels.

Theoretical considerations

We first try to understand the extent of the variation of the data for each of the ten integers (i =0-9). Let j denote the count of the j th image matrix A_{ij} where A is the 28 X 28 greyscale pixel matrix. (j varies from 1 to N_i number of training data for the integer i.)

We can calculate the mean of $\mathbf{A}_j$ and also the standard root mean square of the deviation sigma from the mean,

$$\mathbf{Sigma}_i = ((A_{ij}-\mathbf{A}_j) * (A_{ij}-\mathbf{A}_j) /N_i)^{**} 0.5 \qquad (1)$$

From this, we calculate the sum of the matrix $\mathbf{sumS}_i$ and use $2*\mathbf{sumS}_i$ as the test to determine whether an image belongs to the i th integer.

Next, we test the test-data against these values and establish that no data for a particular integer fails the test.

At this point, we introduce our interpolation function. This is the Lagrange polynomial, which can be written down directly for the x direction as

$$L^n_k(x) = \frac{(x-x_0)(x-x_1)\ldots(x-x_{k-1})(x-x_{k+1})\ldots(x-x_n)}{(x_k-x_0)(x_k-x_1)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_n)} \qquad (2)$$

Where the subscript indicates the x coordinate at the node

And k varies from 0 to n nodes.

The value of the polynomial is 1 at x_k and 0 at all the other nodes, viz the polynomial passes through all the other nodes.

Thus in two dimensions the interpolation matrix NIJ is given by

$$NIJ= L^n_i (x)\, L^m_j (y) \qquad (3)$$

For the i, j nodes in the x, y directions respectively.

In order to use our interpolation functions, we divide our domain into a number of FE meshes. For simplicity, we choose p = q = 4, 6, 14 pixels in the x, y directions because it fits exactly into our 28 X 28 data image.

We note that our quadratic interpolation function only occupies a 3 X 3 position in each mesh. We set the interpolating matrix to the appropriate mean value Ameanij.

For convenience, we can calculate the interpolation matrix for all the pixel positions within the mesh, using s,t as subscripts for the pixels, we can generate interpolating arrays Nstij where ij indicates the interpolating matrix Ameanij discussed above.

Computer Program and results

With the above definitions, we developed a computer program using the Python Language. We calculated the mean values using the training data and tested the interpolating arrays for all the test data. We report that there were no failures.

Table 1 gives the sigma **sumS** and the normalizing sum of the mean values **A**.

Integer test	0	1	2	3	4…
Sigma **sumS**	4.93	2.65	5.16	4.74	4.48
Mean sum	135.8	59.6	116.2	110.8	117.8
Sigma norm	0.036	0.044	0.044	0.042	0.038

Table 1. sums of the mean sigma and mean pixel values from the training data.

Table 1 gives the results obtained for the first five integer fits. We note that the mean deviation in the least squares sense is around 0.04 and does not vary significantly from integer to integer being tested.

Tests were also conducted to see the cross correlations between the integer results. The cross correlations were of the order of 1, normalized for when the integer

values were not equal. This last result means that there should be no false positives.

Discussion of results

The results demonstrate the ability of FE Interpolation to perform in the number recognition problem. Results were good even for the 14 pixel mesh where only 4 elements was used for the whole problem. Here an estimate of the truncation error with h=1/4, would put C h(0)**3 = C /64. Apparently, for our problem C was of the order of less than 1. The results were obtained with a minimum of computing resources.

It would be of interest to compare the process required in the deep learning solution presented by Nielsen [3] using Artificial Neural Networks (ANN) [5]. The input variables consisted of 50,000 data with 784 pixels each. There was a hidden layer of 15 Neurons. In order to appreciate the computing effort required to compute the optimal weights w in the problem, we note that there are 784*15 w for the first layer and 15*10 w for the hidden layer. This is equivalent to solving 12,000 quasi linear problems. There is no fixed opinion as to the role of the hidden layer. In the writer's opinion and because the equations for each layer is quasi linear, the addition of another quasi linear function to the results of the first layer can be regarded as an approximation to a nonlinear function. The solution of the weights is obtained via an iterative back-propagation process [6]. Essentially this is a relaxation process which needs to be more complex because of the non-positive definite nature of the quasi linear problem, at best we can estimate an order of magnitude computing by assuming that each solution requires 12000*15*15 operations to solve a similar linear problem with the same number of unknowns and a bandwidth of the 15 hidden layer of neurons. We must do this 50,000 times. Clearly the effort after the first ten integer results is much reduced because we need only account for changes. However it is safe to characterize this as a global problem requiring the full extent of each input and output variable to be involved. Fig.2 shows the neural network used for this problem.

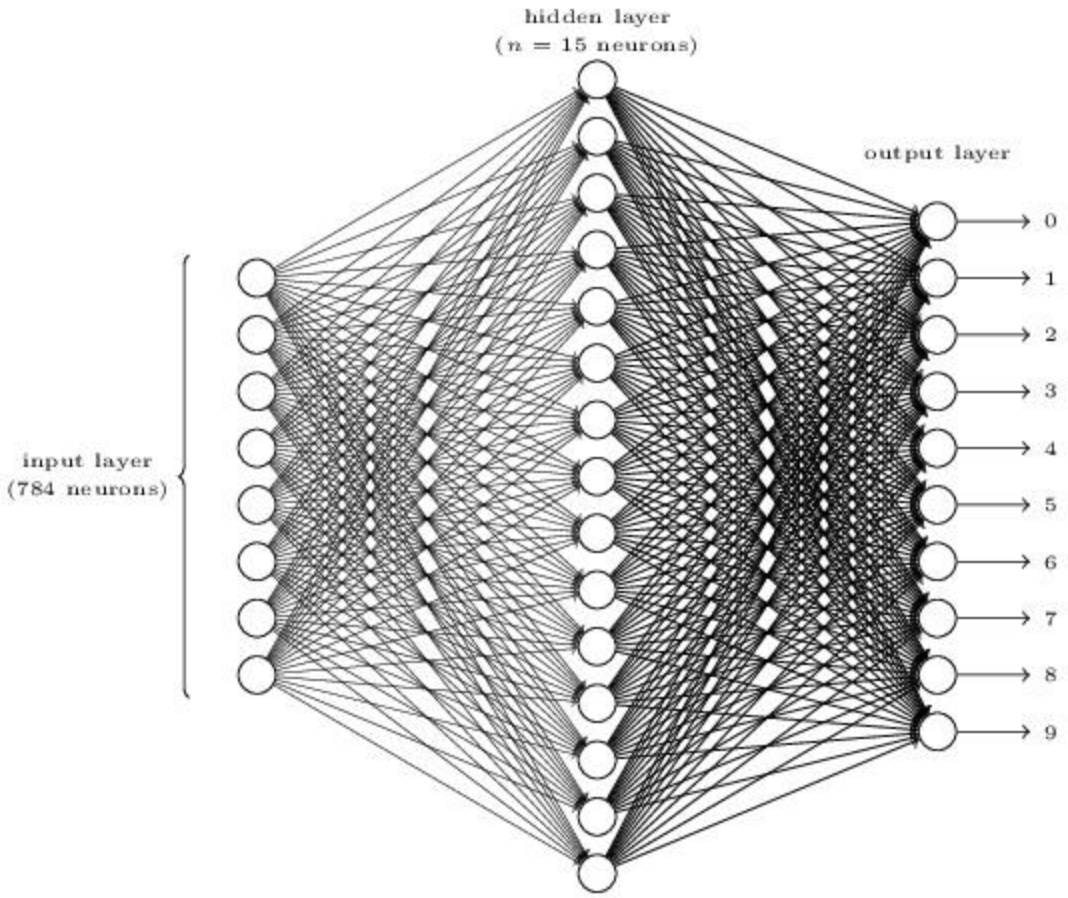

Fig. 2. Neural network with hidden layer for MNIST data.

In contrast, the FE interpolation is a local problem requiring only the mesh pixel. There are linear or nonlinear equations to be solved. It is interesting to speculate that the FE interpolation is the better model for implementation in the brain. The method described here should be applied to further imaging problems, perhaps extending to 3 D problems.

Conclusions

A Least Squares FE Interpolation process was developed and applied to the problem of recognizing hand-written numbers.

1. The FE Interpolation was successfully applied.
2. The quadratic Lagrange polynomial was found to be sufficient for the task.

3. The FE Interpolation was shown to be local and requiring minimal computing resources especially when compared to the deep learning process for this problem.
4. Part of the advantage of the FE process is that the mesh used can be adapted to fit the data.

References

[1] Zienkiewicz, O. C., and Taylor, R. L., The Finite Element Method, 5th ed., Vol. 1: The Basis, Chapter 14 (Errors, recovery processes and error estimates), pp. 365-400. Butterworth-Heinemann (2000).

[2] P.V. Marcal, J.T. Fong, R. Rainsberger, L. Ma, Finite Element Analysis of a Pipe Elbow Weldment Creep- Fracture Problem Using an Extremely Accurate 27-node Tri-Quadratic Shell and Solid Element Formulation, Proc. ASME PVP-2016, Vancouver, Canada, 2016.

[3] Nielsen, M.A., Neural Networks and Deep Learning, Determination Press, 2015.

[4] Yu, Q., The MNIST DATABASE of handwritten digits, 2007.

[5] Rumelhart, D.E., McClelland, J.L. and PDP Research Group, Parallel Distributed Processing, MIT Press, 1988.

[6] Rumelhart, D.E., Hinton, G.E., Williams, R.J., Learning representations by back-propagating errors, *Nature*. **323** (6088): 533–536, October 1986.