Large-Scale Matrix Inversion using neural networks

Machine learning

By Vaishnavi C K

INTRODUCTION

Matrix inversion is used widely in signal processing, robotics and image processing. In real-time problems, order of the matrices are large and thus there is a need for computing large-scale inverse matrices.

In machine learning, recurrent neural network is a type of neural network in which the output from previous step are fed as input to the current step.

For training through RNN:

- A single time step of the input is provided to the network and the current state is calculated using set of current inputs and the previous state. The current state becomes the previous state for the next time step.
- After the completion of all the time steps, final current state is used to calculate the output.
- The output is then compared to the actual output, that is, the target output and the error is generated. The error is then back-propagated to the network to update the weights and hence the network (RNN) is trained.

PROBLEM STATEMENT

Computing the inverse of a higher-order matrix using recurrent neural network.

METHODOLOGY

- To compute the inverse of a large scale nxn non singular matrix, recurrent neural network uses the formula
  $dV(t) / dt = -\eta A^T A\, V(t) + \eta A^T$

- A - input matrix
  V(t) - matrix which has all the elements in the same position corresponding to the inverse matrix A^{-1}.
  η - positive scalar parameter

- In the code , the derivative in the formula $dV(t)/dt = -\eta A^T A\, V(t) + \eta A^T$ is taken in the form of difference.
  The time here is $t = nT$, where T is the sampling instant.

  Implies , $(V(nT) - V((n-1)T)) / T = -\eta A^T A\, V(nt) + \eta A^T$

- Sampling instant is assumed to be 1 ie T = 1. Thus,

$$V(n) = V(n-1) - \eta A^T A\, V(n) + \eta A^T$$

The above equation is used in the code

- V is initialized with 0.01 and the order of V matrix will (mx2n) where,
  m - number of rows of A
  n - number of columns of A

- Instead of indexing and storing the new estimate matrix V(n) and old estimate matrix V(n-1) separately, a single matrix V stores both the new estimate and the old estimate matrix, which is made by stacking 2 matrices side by side is used.

- New estimate matrix V(n) and old estimate matrix V(n-1) are extracted from V by applying slicing operation. The formula is then applied on the new estimate matrix V(n) and this is carried out from n=1 to <some large number> of iterations.

- The new estimate matrix V(n) is stored in the place where the old estimate matrix was there. After the iterations the new estimate of the inverse V(n) is extracted and this is the inverse of the matrix obtained from the RNN.

- Error is obtained by subtracting the inverse obtained and the actual inverse. The error can be reduced by either increasing the number of iterations or by reducing the value of eta.

CODE SUMMARY

```
%
clear;
A= randn(25,25)*100;
s=size(A);
eta=0.0000001;
V=ones(s(1), 2*s(2))/100;
%
```

- A is the input matrix which is of order 25x25 and is initialised with random values.

- s stores the size(or order) of the matrix in the form of an array.
  s = [number of rows number of columns]
  In this case , s= [25 25]
  s(1) = number of rows = 25
  s(2) = number of columns = 25

- eta is chosen arbitrarily as 0.0000001. The value of eta should be small enough to avoid overshooting.

- V is of order 25x50 in this case and its values are initialized as 0.01.

```
%
for n=1:1000000

   V( :, 1 : s(1) ) = V( :,  s(2)+1 :  2*s(2) )   -  eta* A‘ *A* V( :,  1 : s(1)  ) + eta* A‘ ;

   V( :,  s(2)+1 : 2*s(2) ) = V( :, 1 : s(1) );

end
%
```

- n is iterated from 1 to 1000000. The required update is carried out for every iteration.
- Here V(n) and V(n-1) are extracted from V by slicing operation.
- V(:, 1 : s(1)) is V (:, 1: 25) as s(1) = 25 . Thus all rows are extracted and the columns are extracted from 1 to 25 . This is V(n)
- V(:, s(2)+1 : 2*s(2)) is V(:, 26 : 50) as s(2)=25 . Thus all rows are extracted and the columns are extracted from 26 to 50. This is V(n-1).
- New estimate V(n) is updated as per the formula $\mathbf{V(n) = V\,(n-1) - \eta A^T A\ V(n) + \eta A^T}$ and this value is stored in the place of the old estimate V(n-1) .

```
%
V( :, 1 : s(1) )
g = inv(A)
error= V( :, 1 : s(1) ) - inv(A)
%
```

- V(:, 1 : s(1)) is the new estimate of the inverse, V(n) which is obtained after the iteration . This is the inverse obtained from rnn.

Inbuilt functions used in the code:

- randn- to generate arrays of random numbers
- size- returns the order/ size of the matrix
- ones- returns a matrix with values equal to 1
- slicing operation- to generate sub matrices

RESULT

For a 5x5 matrix:

Input matrix A

A | ans | g | error | V

5x5 double

	1	2	3	4	5
1	61.1141	75.7302	-57.8970	-146.9244	114.7856
2	-33.0262	-18.5805	170.0212	125.1590	-25.3076
3	19.0535	-29.6454	4.8454	-96.2664	-81.8371
4	80.8242	33.0223	2.1769	11.3565	-56.5512
5	-66.7199	-17.3930	-40.4053	5.6525	13.7189

V matrix

A | ans | g | error | V

5x10 double

	1	2	3	4	5	6	7	8	9	10
1	-0.0066	-0.0091	-0.0046	-0.0098	-0.0298	-0.0066	-0.0091	-0.0046	-0.0098	-0.0298
2	0.0148	0.0149	0.0028	0.0299	0.0433	0.0148	0.0149	0.0028	0.0299	0.0433
3	0.0034	0.0067	0.0034	-0.0013	-8.9898e-04	0.0034	0.0067	0.0034	-0.0013	-8.9898e-04
4	-0.0044	-0.0022	-0.0067	8.0356e-04	-0.0039	-0.0044	-0.0022	-0.0067	8.0356e-04	-0.0039
5	-0.0015	-0.0045	-0.0062	-0.0142	-0.0181	-0.0015	-0.0045	-0.0062	-0.0142	-0.0181

V(n) (estimate inverse obtained from RNN)

A | ans | g | error | V

5x5 double

	1	2	3	4	5
1	-0.0066	-0.0091	-0.0046	-0.0098	-0.0298
2	0.0148	0.0149	0.0028	0.0299	0.0433
3	0.0034	0.0067	0.0034	-0.0013	-8.9898e-04
4	-0.0044	-0.0022	-0.0067	8.0356e-04	-0.0039
5	-0.0015	-0.0045	-0.0062	-0.0142	-0.0181

Error

A | ans | g | error | V

5x5 double

	1	2	3	4	5
1	8.4817e-12	9.5526e-12	3.8785e-12	1.7493e-11	2.8745e-11
2	-1.4596e-11	-1.6440e-11	-6.6746e-12	-3.0106e-11	-4.9471e-11
3	-3.9150e-13	-4.4092e-13	-1.7923e-13	-8.0698e-13	-1.3264e-12
4	1.2875e-12	1.4499e-12	5.8901e-13	2.6545e-12	4.3624e-12
5	6.0462e-12	6.8096e-12	2.7650e-12	1.2470e-11	2.0491e-11

Actual inverse

A | ans | g | error | V

5x5 double

	1	2	3	4	5
1	-0.0066	-0.0091	-0.0046	-0.0098	-0.0298
2	0.0148	0.0149	0.0028	0.0299	0.0433
3	0.0034	0.0067	0.0034	-0.0013	-8.9898e-04
4	-0.0044	-0.0022	-0.0067	8.0356e-04	-0.0039
5	-0.0015	-0.0045	-0.0062	-0.0142	-0.0181

Conclusion and Future Scope

- The convergence rate of the neural network is independent of the order of the matrices. This feature renders the neural network an advantage over traditional sequential procedures for large-scale matrix inversion.

- Since the convergence rate can be controlled by properly selecting the scaling contrast 7, the recurrent neural network is also advantageous in real-time applications.

- Thus the recurrent neural network method for computing inverse of large matrices is highly efficient and further research based on this method can be aimed at extension to computing pseudo inverse and implementation in analog VLSI circuits.

Reference

Jun Wang,”A recurrent neural network for real-time matrix inversion”, Applied Mathematics and Computation, Volume 55, Issue 1, 1993