# Ill-Conditioned 정방행렬의 단측 역행렬 산출용 유사 인공신경망 알고리듬

문병수, 양성운, 김영택
한국원자력연구소

# A Neural-like Algorithm to Compute One-Sided Inverse of Ill-Conditioned Matrices

Byung Soo Moon, Sung Woon Yang, Young Taek Kim
Korea Atomic Energy Research Institute

**요 약**

이 논문에서는 크기가 큰 Ill-Conditioned 정방행렬의 좌측 또는 우측 역행렬 계산시 계산상의 정확도를 향상시키는 알고리듬에 대하여 기술한다. 이 알고리듬은 대상 행렬의 행 벡터들을 Input으로 하고 해당 Input 벡터가 몇번째 행 벡터인지를 나타내는 단위 벡터를 Target 벡터로 하며 초기 Weight 값으로 Pivoting을 겸한 Gauss소거법을 적용하여 얻은 역행렬을 사용하는 Single Layer 인공신경망에 적용하는 역전파 알고리듬과 흡사한 것이다. 각각의 Input 행 벡터에 대하여 역행렬의 열 벡터들이 점진적으로 직교가 되거나 평행이 되도록 근접시키므로써 모든 Input 행 벡터들이 열벡터들에 비교적 균일하게 직교 또는 평행이 되도록 학습시키는 알고리듬이다.

## 1. 서 론

정칙 정방행렬의 역행렬 산출에 관한 최근 연구 결과 발표는 LU Decomposition을 사용하는 방법에 대한 이론적인 연구[1]나 대칭행렬에 관한 결과[2] 등으로 나타나 있다. 이 논문에서는 임의의 Ill-Conditioned행렬의 역행렬에 대한 계산상의 정확도 향상을 위한 실용적인 알고리듬에 대하여 기술하고자 한다. 이 논문에서는 벡터의 크기와 행렬의 Norm으로 Supremum Norm을 사용한다. 즉, n-벡터 $x$의 경우

$$||x||_\infty = \max\{|x_i| \mid j=1,2,\cdots,n\}$$

을 사용하며, 행렬 $A$의 Norm은

$$||A||_\infty = \max\left[\frac{||Ax||_\infty}{||x||_\infty} \mid x\neq 0\right]$$

을 사용한다. 또한, 행렬 $A$의 Condition Number는 $K=||A||_\infty||A^{-1}||_\infty$ 형태를 사용한다.

## 2. 단측 역행렬 산출 알고리듬

임의의 정칙인 n차 정방행렬을 $A$라 하고 Scaled Partial Pivoting과 같은 Pivoting 알고리듬과 함께 가우스 소거법을 적용하여 얻은 $A$ 의 역행렬을 $A^{-1}$ 라 하자. 만약, $\xi_j$를 역행렬 $A^{-1}$의 $j$번째 열벡터라 하고 $e_j$ 를 $j$번째 원소가 1인 단위 벡터라 하면 $j=1,2,\cdots,n$에 대하여

$$A\xi_j = e_j + \delta_j \qquad \text{------ (1)}$$

가 성립한다. 여기서, $\delta_j$는 오차를 나타내는 벡터로서 필요시 Double Precision을 사용하고 가우스 소거법에 의한 $A^{-1}$산출시 적당한 인자를 곱한 후 계산함으로써 $||\delta_j||_\infty < 1$이라 가정할 수 있다.

이상적으로는 $A^{-1}$의 열벡터 $\xi_j$대신 $A\xi_j' = e_j$를 만족하는 $\xi_j'$를 구하고자 한다. 즉, $A(\xi_j' - \xi_j) = -\delta_j$를 만족하는 $\xi_j'$에 근접하는 벡터를 산출하여 $\xi_j$를 대체코자 한다. 만약, $B$를 $A$의 좌측 역행렬로서 가우스

소거법에 의하여 산출한 $A^{-1}$이나 $A^T$에 가우스 소거법을 적용하여 얻은 역행렬의 전치행렬로 두면 $\xi_j'$는 $\xi_j' = \xi_j - W\theta_j$로 나타낼 수 있다.

임의의 양의 실수 $\lambda$를 $\lambda\|W\|_\infty < 1$ 을 만족하도록 취하고

$$\xi_j^{(k+1)} = \xi_j^k - \lambda W\theta_j = \xi_j^k - \lambda W(A\xi_j - e_j) \quad \text{-- (2)}$$

로 두어 $k=0,1,2,\ldots$ 에 대하여 반복연산을 수행하면 생성된 벡터 열 $\{\xi_j^k \mid k=0,1,2,\ldots\}$은 수렴하며 $\xi_j'$에 근접함을 알수 있다.

위 식 (2)는 $j=1,2,3,\ldots n$ 등 n 개의 벡터 $\xi_j$에 대한 반복연산으로 이들을 행렬 형태로 표현하면

$$A_{k+1}^{-1} = A_k^{-1} - \lambda W(AA_k^{-1} - I) \quad \text{---- (3)}$$

가 된다. 여기서 $I$는 단위행렬이다. 식(3)을 사용 반복연산을 수행함으로써 정방행렬 $A$의 우측 역행렬에 대한 정확도를 높일 수 있다.

이제 식(3)의 연산이 역전파 알고리듬(Back Propagation Algorithm)과 어떻게 흡사한지를 보기 위하여 식(3)의 우변 둘째 항에 관계식 $I = \sum_{j=1}^{n} e_j e_j^T$를 사용하면

$$\lambda W(AA_k^{-1} - I) = \lambda W\left(\sum_{j=1}^{n} e_j e_j^T\right)(AA_k^{-1} - I)$$

$$= \lambda \sum_{j=1}^{n} \zeta_j (a_j^T A_k^{-1} - e_j^T)$$

로 표현할 수 있다. 여기서, $\zeta_j$는 행렬 $W$의 $j$번째 열 벡터이며 $a_j$는 행렬 $A$의 $j$번째 행벡터이다. 따라서, 식 (3)은

$$A_{k+1}^{-1} = A_k^{-1} - \lambda \sum_{j=1}^{n} \zeta_j (a_j^T A_k^{-1} - e_j^T) \quad \text{--- (4)}$$

가 된다.

식 (4)를 사용하여 $A_k^{-1}$로 부터 $A_{k+1}^{-1}$을 산출하는 연산은 $j=1,2,\ldots n$에 대한 n개의 연산

$$A_{m+1}^{-1} = A_m^{-1} - \lambda\zeta_j (a_j^T A_m^{-1} - e_j^T) \quad \text{---- (5)}$$

과 흡사함을 알수 있다. 뿐만아니라, 식(5)는 $a_j$를

$$A = \begin{bmatrix}
1.00000 & -0.05000 & 0.00167 & 0.00000 & \ldots \\
1.00000 & 0.00000 & -0.00083 & 0.00000 & \ldots \\
1.00000 & 0.05000 & 0.00167 & 0.00000 & \ldots \\
1.00000 & 0.10000 & 0.00917 & 0.00075 & \ldots \\
1.00000 & 0.15000 & 0.02167 & 0.00300 & \ldots \\
1.00000 & 0.20000 & 0.03917 & 0.00750 & \ldots \\
1.00000 & 0.25000 & 0.06167 & 0.01500 & \ldots \\
1.00000 & 0.30000 & 0.08917 & 0.02625 & \ldots \\
1.00000 & 0.35000 & 0.12167 & 0.04200 & \ldots \\
1.00000 & 0.40000 & 0.15917 & 0.06300 & \ldots \\
1.00000 & 0.45000 & 0.20167 & 0.09000 & \ldots \\
1.00000 & 0.50000 & 0.24917 & 0.12375 & \ldots \\
1.00000 & 0.55000 & 0.30167 & 0.16500 & \ldots \\
1.00000 & 0.60000 & 0.35917 & 0.21450 & \ldots \\
1.00000 & 0.65000 & 0.42167 & 0.27300 & \ldots \\
1.00000 & 0.70000 & 0.48917 & 0.34125 & \ldots \\
1.00000 & 0.75000 & 0.56167 & 0.42000 & \ldots
\end{bmatrix}$$

Input 벡터로 하고 $e_j$를 Target 벡터로 하며 Weight Array 가 $A_m^{-1}$인 Single Layer 신경망에 역전파 알고리듬을 적용하는 알고리듬으로 간주할 수 있다.

연산 수행 결과 얻게 되는 Weight Array는 구하고자 하는 우측 역행렬의 근사해가 된다. 이때 Forward 연산 출력 OUTPUT은 $a_j^T A_m^T - e_j^T$가 되며 Reverse연산에서 Weight의 수정값은 $\lambda\zeta_j \times OUTPUT$ 이 된다. 여기서 $\zeta_j$는 기존의 Weight와 Target vector $e_j$의 곱에 의하여 산출된다.

다음 절 예제에서 확인된바에 의하면 역전파 인공신경망 알고리듬 (5)보다는 식(4)의 반복 연산에 의한 우측 역행렬 산출 결과에 대한 $\|AA^{-1} - I\|_\infty$의 값이 더 작음을 알수 있다.

### 3. 알고리듬 적용 예

이절에서는 앞서 기술한 알고리듬을 Condition

Number가 큰 한 행렬에 적용한 결과에 대하여 기술한다. 표에 나타난 행렬 A는 구간 [0, 1]에서 Discrete Point형태로 주어진 임의의 함수를 다항식 형태의 함수로 "Function Identification"을 수행하는 과정에서 발생되는 23차 정방행렬의 첫 6개열의 일부이다.

$$A^{-1}=\begin{bmatrix}
+0.167E+00 & +0.667E+00 & +0.167E+00 & -0.114E-06 & +0.249E-06 & \cdots\\
-0.100E+02 & +0.874E-15 & +0.100E+02 & -0.153E-15 & -0.451E-15 & \cdots\\
+0.247E+03 & -0.686E+03 & +0.107E+04 & -0.188E+04 & +0.325E+04 & \cdots\\
-0.336E+04 & +0.169E+05 & -0.462E+05 & +0.990E+05 & -0.178E+06 & \cdots\\
+0.278E+05 & -0.203E+06 & +0.763E+06 & -0.199E+07 & +0.392E+07 & \cdots\\
-0.145E+06 & +0.146E+07 & -0.703E+07 & +0.215E+08 & -0.465E+08 & \cdots\\
+0.449E+06 & -0.670E+07 & +0.404E+08 & -0.142E+09 & +0.335E+09 & \cdots\\
-0.547E+06 & +0.191E+08 & -0.148E+09 & +0.595E+09 & -0.154E+10 & \cdots\\
-0.161E+07 & -0.270E+08 & +0.329E+09 & -0.157E+10 & +0.456E+10 & \cdots\\
+0.930E+07 & -0.186E+08 & -0.310E+09 & +0.229E+10 & -0.819E+10 & \cdots\\
-0.213E+08 & +0.167E+09 & -0.403E+09 & -0.626E+09 & +0.697E+10 & \cdots\\
+0.250E+08 & -0.319E+09 & +0.158E+10 & -0.384E+10 & +0.324E+10 & \cdots\\
+0.483E+07 & +0.184E+09 & -0.194E+10 & +0.879E+10 & -0.240E+11 & \cdots\\
-0.109E+09 & +0.364E+09 & +0.251E+10 & -0.233E+11 & +0.904E+11 & \cdots
\end{bmatrix}$$

구간 [0, 1]을 20등분하고 부분구간들의 끝점인 21개 점에서의 함수값과 좌우 두 끝점 0 과 1 에서의 도함수의 값이 $P_m(x)=x^m$의 대응하는 값과 같도록 3차 Spline 보간함수를 구하면 $m=0,1,2,\cdots,22$에 대한 계수들은 행렬 $A$와 같다. 즉, 행렬 $A$의 첫 번째 열은 $P_0(x)=1$에 대한 3차 Spline계수 들이며 둘째 열은 $P_1(x)=x$에 대한 계수들이다. 행렬 A 에 대한 $||A||_\infty$의 값을 구하면 38.5로 비교적 작은 값이지만 Scaled Partial Pivoting방법과 함께 가우스 소거법에 의하여 구한 역행렬 $A^{-1}$에 대한 Norm값 $||A^{-1}||_\infty$은 $0.12\times10^{14}$로 매우 큰 값이다. 따라서, 행렬 $A$의 Condition Number 는 약 $0.46\times10^{15}$이 되며 행렬 $A$는 Ill-Conditioned 임을 알 수 있다.

가우스 소거법에 의하여 산출한 역행렬 $A^{-1}$을 행렬 $A$의 우측에 곱한 다음 단위 행렬과의 차이의 Norm을 구하면, 즉 $||AA^{-1}-I||_\infty$의 값을 구하면 0.0018이나 본 논문에서 제안한 알고리듬을 적용한 후의 값은 0.00017 로 오차의 값이 약 10분의 1 수준으로 감소 되었음을 확인 할 수 있다.

## 4. 결론

이상에서는 주어진 정칙 정방행렬의 우측 역행렬에 대한 계산상의 정확도를 향상시키는 알고리듬에 대하여 기술하였다. 좌측 역행렬의 경우에는 식(1)을

$$\xi_j^T A=e_j^T+\delta_j^T \qquad \text{--------} \quad (1')$$

에 의하여 대체한 후 동일한 방법을 적용하면 된다. 이때 $\xi_j$는 $A^{-1}$의 j번째 행 벡터가 된다.

한편, 식( 1')는

$$A^T\xi_j=e_j+\delta_j \qquad \text{----------} \quad (6)$$

이 되므로 행렬 $A$대신 $A^T$의 우측 역행렬을 산출한 후 전치행렬을 취하면 동일한 알고리듬에 의하여 행렬 $A$의 좌측 역행렬을 구할 수 있음을 확인할 수 있다.

## 참고문헌

[1] K. Kubota, "Matrix Inversion Algorithms by Means of Automatic Differentiation", Appl. Math. Lett. Vol.7, No.4 (1994) 19-22.

[2] S. A. Shishkin, "Extending the Potentialities of Matrix Inversion Procedures", Comput. Maths. Math. Physics, Vol.31, No.4 (1991) 7-16.

[3] P. M. Prenter, Splines and Variational Methods", John Wiley & Sons (1979).