韓國컴퓨터情報學會 論文誌
第14卷 第8號, 2009. 8.

2009-14-8-1-2

Jacobian 행렬의 주부분 행렬을 이용한 Levenberg-Marquardt 알고리즘의 개선

곽 영 태*, 신 정 훈*

Improving Levenberg-Marquardt algorithm using the principal submatrix of Jacobian matrix

Young-tae Kwak*, Jung-hoon Shin*

요 약

본 논문은 Levenberg-Marquardt 알고리즘에서 Jacobian 행렬의 주부분 행렬을 이용하여 학습속도를 개선하는 방법을 제안한다. Levenberg-Marquardt 학습은 오차함수에 대한 2차 도함수를 계산하기 위해 Hessian 행렬을 사용하는 대신 Jacobian 행렬을 이용한다. 이런 Jacobian 행렬을 가역행렬로 만들기 위해, Levenberg-Marquardt 학습은 μ값을 증가시키거나 감소시키는 과정을 수행하고 μ값의 변경에 따른 역행렬의 재계산이 필요하다. 따라서 본 논문에서는 μ값의 설정을 위해 Jacobian 행렬의 주부분 행렬을 생성하고 주부분 행렬의 고유값 합을 이용하여 μ값을 설정한다. 이와 같은 방법은 추가적인 역행렬 계산을 하지 않으므로 학습속도를 개선할 수 있다. 제안된 방법은 일반화된 XOR 문제와 필기체 숫자인식 문제를 대상으로 실험하여 학습속도의 향상을 검증하였다.

Abstract

This paper proposes the way of improving learning speed in Levenberg-Marquardt algorithm using the principal submatrix of Jacobian matrix. The Levenberg-Marquardt learning uses Jacobian matrix for Hessian matrix to get the second derivative of an error function. To make the Jacobian matrix an invertible matrix, the Levenberg-Marquardt learning must increase or decrease μ and recalculate the inverse matrix of the Jacobian matrix due to these changes of μ. Therefore, to have the proper μ, we create the principal submatrix of Jacobian matrix and set the μ as the eigenvalues sum of the principal submatrix, which can make learning speed improve without calculating an additional inverse matrix. We also showed that our method was able to improve learning speed in both a generalized XOR problem and a handwritten digit recognition problem.

▸ Keyword : 오류역전파학습, Levenberg-Marquardt 학습, Jacobian 행렬

• 제1저자 : 곽영태
• 투고일 : 2009. 04. 27, 심사일 : 2009. 05. 24, 게재확정일 : 2009. 08. 07.
* 전북대학교 응용시스템공학부

Ⅰ. 서 론

신경회로망은 입력과 출력에 따라 행동을 결정하는 특성 때문에 패턴인식, 함수 근사화, 최적화 이론 등에 많이 사용되고 있으며 특히, 개인용 컴퓨터의 발전과 더불어 비약적인 발전을 이루고 있다.

입력과 목표패턴의 값에 따라 가중치를 결정하는 교사학습에 사용되는 퍼셉트론 학습규칙은 입력패턴의 선형분리라는 제한 조건을 가지고 있었다. 그러나 David Rumellhart[1]에 의해 발전된 오류역전파학습(Error Backpropagation)은 단층퍼셉트론의 LMS(Least Mean Square)학습을 다층퍼셉트론으로 일반화하여 많은 응용 분야에 적용되고 있다[2,3]. 그러나 이런 오류역전파학습은 활성화 함수의 비선형적인 특성 때문에 오차함수의 굴곡이 심하고 지역최소점을 가질 수 있어 학습속도가 느린 단점이 있다. 다층퍼셉트론의 또 다른 학습으로는 은닉층의 활성화 함수를 RBF(Radial Basis Function)로 사용하고 층력층을 LMS로 학습하는 방법이 있다[4]. RBF는 학습패턴이 클러스터가 용이한 응용에 적합하다.

지금까지 오류역전파학습의 학습속도를 향상시키기 위하여 여러 가지 방법이 제안되었는데, 첫 번째는 경험적인 방법으로, 모멘텀 항을 첨가하는 방법, 학습률을 변화시키는 방법[5], 변수의 스케일을 조정하는 방법[6] 등이 있었다. 그리고 최근에는 활성화함수의 기울기를 조정하는 방법[7] 등이 있었다. 그러나 이런 방법들은 추가적인 파라미터들을 요구하고 복잡한 문제의 경우에는 파라미터의 선택에 따라 성능이 좌우되는 경우도 발생하고 있다.

두 번째는 최적화 이론에서 사용하는 수치적인 방법으로 오차함수에 대한 2차 도함수를 사용하는 방법이다. 이런 방법으로는 공액 변화율(Conjugate Gradient) 알고리즘[8,9]과 Newton방법의 변종인 LM(Levenberg-Marquardt)알고리즘[10]이 있다. 그리고 학습속도를 개선하기 위하여 Costa[11]는 가중치 벡터의 놈(norm)을 제한하는 방법을 제안하였다. 이런 방법 모두 2차 도함수인 Hessian 행렬을 구하는 대신 근사적인 방법으로 2차 도함수를 구하여 학습에 적용함으로써 빠른 학습속도를 보인다.

그러나 공액 변화율 알고리즘은 오차함수가 완전한 2차 함수가 아니기 때문에 선형 검색을 대신할 방법을 찾아야 하며 매번 학습시 초기화를 시켜야 하는 문제점이 있다[12]. 반면에 LM학습은 Hessian 행렬을 구하는 대신 Jacobian 행렬을 이용하여 근사적으로 2차 도함수를 구하고 있으나 Jacobian 행렬을 저장하고 역행렬을 구해야하는 단점이 있다. 그러나 다층신경회로망의 크기가 아주 크지 않은 경우 LM학습이 다른 학습방법에 비해 빠르다고 평가되고 있다[13].

이런 LM학습에서 μ값은 중요한 역할을 하는데, μ의 값이 크면 오류역전파학습과 유사해지며, 작으면 Gauss Newton 방법의 수렴속도를 가진다. 또한, 가중치 조정에 따른 오차함수의 변화에 따라 μ를 확대하거나 축소해야 한다. 즉, LM학습을 적용하는 응용에 따라 μ의 최대값, 최소값을 정해야하는 문제점이 발생한다. 따라서 본 논문에서는 Jacobian 행렬($\mathbf{J}^T\mathbf{J}$)의 고유값 합을 이용하여 근사적인 의 μ 값을 결정하는 방법을 제시한다.

논문의 구성은 2장에서는 다층퍼셉트론의 구성과 오류역전파학습, LM학습 알고리즘을 설명하고 개선된 LM학습 알고리즘을 제시한다. 3장에서는 간단한 일반화된 XOR문제와 필기체숫자 인식문제를 통하여 제안된 방법을 실험하고 4장에서 결론을 맺는다.

Ⅱ. 본론

2장에서는 다층퍼셉트론의 구조, 오류역전파학습, LM학습, 개선된 LM학습을 기술한다.

2.1 다층퍼셉트론과 오류역전파학습

일반적인 다층퍼셉트론은 그림 1과 같으며 논문에서는 은닉층이 하나 있는 구조를 사용한다. 학습을 위한 입력 및 목표패턴은 수식(2.1)과 같고, $k+1$층의 i번째 노드의 출력은 수식(2.2)와 같이 계산된다.

$$\{(\underline{\mathbf{p}}_1, \underline{\mathbf{t}}_1), (\underline{\mathbf{p}}_2, \underline{\mathbf{t}}_2), \cdots, (\underline{\mathbf{p}}_Q, \underline{\mathbf{t}}_Q)\} \quad \cdots\cdots (2.1)$$

$$a^{k+1}(i) = f^{k+1}(n^{k+1}(i)) \quad \cdots\cdots (2.2)$$

$$n^{k+1}(i) = \sum_{j=1}^{Sk} w^{k+1}(i,j)a^k(j) + b^{k+1}(i)$$

다층퍼셉트론의 최종 출력을 행렬로 표현하면 수식(2.3)이 되고, 각 식에서 언더바는 변수가 벡터임을 나타낸다.

$$\underline{\mathbf{a}}^0 = \underline{\mathbf{p}} \quad \cdots\cdots (2.3)$$

$$\underline{\mathbf{a}}^{k+1} = \underline{f}^{k+1}(\mathbf{W}^{k+1}\underline{\mathbf{a}}^k + \underline{\mathbf{b}}^{k+1})$$

$$k = 0, 1, \cdots, M-1$$

오차함수는 수식(2.4)와 같이 정의하며, 하나의 입력/목표패턴($\underline{\mathbf{p}}_q, \underline{\mathbf{t}}_q$)에 대한 오차함수는 $\hat{E}$로 근사화 된다.

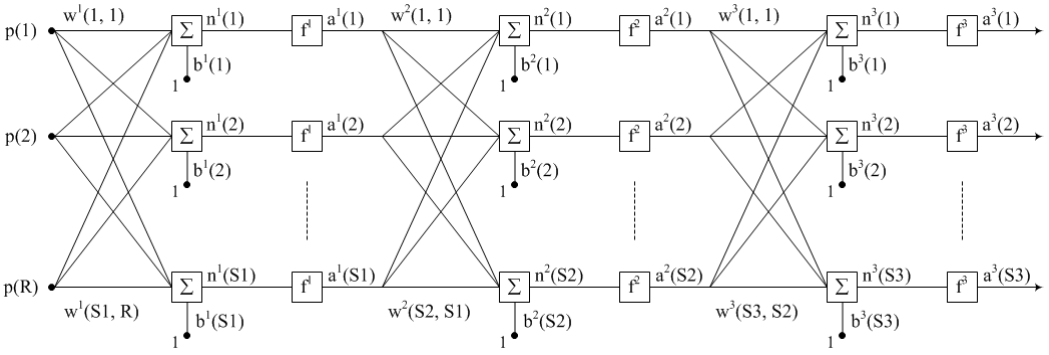


그림 1. 다층퍼셉트론 구조
Fig 1. Structure of Multilayer Perceptrons

$$E(\mathbf{w}) = \frac{1}{2}\sum_{q=1}^{Q}(\underline{\mathbf{t}}_q - \underline{\mathbf{a}}_q^M)^T(\underline{\mathbf{t}}_q - \underline{\mathbf{a}}_q^M) = \frac{1}{2}\sum_{q=1}^{Q}\underline{\mathbf{e}}_q^T\underline{\mathbf{e}}_q$$
$$\hat{E}(\mathbf{w}) = \frac{1}{2}\underline{\mathbf{e}}_q^T\underline{\mathbf{e}}_q \quad \cdots\cdots (2.4)$$

오차함수의 1차 도함수인 최급강하법을 이용하는 오류역전파학습은 가중치와 바이어스의 조정을 위해 수식(2.5)와 수식(2.6)을 이용하며, 수식(2.7)에서 δ^k는 각 노드의 오차신호(error signal)로 마지막 출력층에서 역전파된다. 그리고 수식(2.8)은 마지막 출력층의 오차신호로써 초기 오차신호를 계산하기 위해서 사용된다.

$$\triangle w^k(i,j) = -\alpha\frac{\partial\hat{E}}{\partial w^k(i,j)} \quad \cdots\cdots (2.5)$$
$$\frac{\partial\hat{E}}{\partial w^k(i,j)} = \frac{\partial\hat{E}}{\partial n^k(i)}\frac{\partial n^k(i)}{\partial w^k(i,j)} = \delta^k(i)a^{k-1}(j)$$

$$\triangle b^k(i) = -\alpha\frac{\partial\hat{E}}{\partial b^k(i)} \quad \cdots\cdots (2.6)$$
$$\frac{\partial\hat{E}}{\partial b^k(i)} = \frac{\partial\hat{E}}{\partial n^k(i)}\frac{\partial n^k(i)}{\partial b^k(i)} = \delta^k(i)$$

$$\underline{\boldsymbol{\delta}}^k = \dot{F}^k(\underline{\mathbf{n}}^k)(\mathbf{W}^{k+1})^T\underline{\boldsymbol{\delta}}^{k+1} \quad \cdots\cdots (2.7)$$
$$\dot{F}^k(\underline{\mathbf{n}}^k) = \begin{bmatrix} \dot{f}^k(n^k(1)) & 0 & \cdots & 0 \\ 0 & \dot{f}^k(n^k(2)) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dot{f}^k(n^k(S^k)) \end{bmatrix}$$
$$\text{where} \quad \dot{f}^k(n^k(n)) = \frac{df^k(n)}{dn}$$

$$\underline{\boldsymbol{\delta}}^M = \dot{F}^M(\underline{\mathbf{n}}^M)(\underline{\mathbf{t}}_q - \underline{\mathbf{a}}_q) \quad \cdots\cdots (2.8)$$

이와 같은 오류역전파학습은 오차함수의 1차도함수를 사용하기 때문에 학습속도가 느리고, 지역최소점에 빠질 확률이 높다.

2.2. Levenberg-Marquardt 알고리즘

오류역전파학습이 1차 도함수를 사용하는 반면, LM학습은 Newton 방법을 근사화하는 방법에 적용한다. Newton 방법은 오차함수를 테일러 전개로 근사화하고 2차 미분을 통하여 근사 최소점을 찾는 방법인데, 2차 미분을 구하기 위하여 Hessian 행렬을 사용한다. 그러나 Hessian 행렬을 구하기 위하여 많은 저장공간과 계산시간을 필요로 한다. 따라서 LM학습은 Hessian 행렬을 근사화하는 Jacobian 행렬을 이용한다.

LM학습에서 오차함수는 수식(2.4)처럼 정의되고, Newton 방법에 의한 가중치 조정은 수식(2.9)와 같다.

$$\mathbf{w} = -\left[\nabla^2 E(\mathbf{w})\right]^{-1}\nabla E(\mathbf{w}) \quad \cdots\cdots (2.9)$$

여기서 $\nabla^2 E(\mathbf{w})$는 Hessian 행렬이며 $\nabla E(\mathbf{w})$는 기울기 벡터이다.

LM학습은 Hessian 행렬의 계산 비용을 줄이기 위해 Jacobian 행렬을 근사적으로 사용하여 수식(2.11)과 같이 가중치를 조정한다. 여기서 $\mathbf{I}$는 단위행렬이며 $\mathbf{J}$는 Jacobian 행렬이다.

$$\nabla E(\mathbf{w}) = \mathbf{J}^T(\mathbf{w})e(\mathbf{w}) \quad \cdots\cdots (2.10)$$
$$\nabla^2 E(\mathbf{w}) = \mathbf{J}^T(\mathbf{w})\mathbf{J}(\mathbf{w})$$

$$\triangle\mathbf{w} = -\left[\mathbf{J}^T(\mathbf{w})\mathbf{J}(\mathbf{w}) + \mu\mathbf{I}\right]^{-1}\mathbf{J}^T(\mathbf{w})e(\mathbf{w}) \quad \cdots\cdots (2.11)$$

수식(2.11)에서 파라미터 μ는 $\mu > 0$이고, $E(\mathbf{w})$가 증가하면 μ는 수식(2.12)와 같이 $\beta(>1)$를 곱하고, $E(\mathbf{w})$가 감소하면 β로 나눈다. 따라서 큰 μ값은 오류역전파학습을 수행하며, 작은 μ값은 Gauss Newton 방법을 수행한다.

$$\mu = \begin{cases} \mu\cdot\beta & \text{if } E(\mathbf{w}_{k+1}) > E(\mathbf{w}_k) \\ \mu/\beta & \text{if } E(\mathbf{w}_{k+1}) \le E(\mathbf{w}_k) \end{cases} \quad \cdots\cdots (2.12)$$

이런 LM학습은 학습패턴과 다층퍼셉트론의 가중치가 증가함에 따라 계산 비용이 많이 소요되기 때문에 작은 크기의

다층퍼셉트론의 학습에 유용하다.

Hangan[10,13]은 μ값을 초기에는 작은 값(0.01)으로 설정한 후 학습이 진행됨에 따라 $\beta(10)$을 오차함수의 변화에 따라 곱하거나, 나누는 과정을 반복한다. 그리고 μ값이 최대값에 도달하면 학습을 중단한다. 따라서 μ값의 설정에 따라 학습속도가 달라지므로 본 논문에서는 다음 장에서와 같이 Jacobian 행렬의 주부분 행렬을 이용하여 μ값을 결정하고 학습속도를 개선하고자 한다.

2.3. 개선된 Levenberg-Marquardt 알고리즘

테일러 전개에서 2차 도함수인 Hessian 행렬을 근사화하는 행렬을 구하기 위해서는 수식(2.11)의 역행렬 부분인 수식(2.13)의 행렬 G가 가역행렬이 되어야한다.

$$\mathbf{G}=\mathbf{J}^T\mathbf{J}+\mathbf{I}=\mathbf{H}+\mu\mathbf{I} \quad \cdots\cdots (2.13)$$

수식(2.13)에서 행렬 H의 고유값과 고유벡터를 각각 $\{\lambda_1, \lambda_2, \cdots, \lambda_n\}$과 $\{\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_n\}$라고 가정하면, 수식(2.14)에서와 같이 G의 고유벡터는 H의 고유벡터와 같고, G의 고유값은 $(\lambda_i+\mu)$이다.

$$\mathbf{G}\mathbf{z}_i=\mathbf{H}\mathbf{z}_i+\mu\mathbf{z}_i=\lambda_i\mathbf{z}_i+\mu\mathbf{z}_i=(\lambda_i+\mu)\mathbf{z}_i \quad \cdots\cdots (2.14)$$

행렬 G가 가역행렬이 되고 근사 최소값을 가지기 위해서는 두 가지 조건을 만족해야 한다. 여기서, 근사 최소값이란 수식(2.4)의 오차함수를 테일러 전개로 2차 도함수까지 근사화 했을 때 생기는 이차형식(quadratic form)의 최소값을 의미한다. 우선, 모든 i에 대해 $(\lambda_i+\mu)\neq 0$이어야 하며 행렬 G가 양의 정부호(positive definite)가 되어야 한다. 따라서 모든 i에 대해 $(\lambda_i+\mu)>0$가 되어야 하며 μ가 수식(2.15)의 조건을 만족하면 행렬 G는 가역행렬이 되며 근사 최소값을 가질 수 있다.

$$\mu > \left|\min\lambda_i\right| \quad \cdots\cdots (2.15)$$
$$i=\{\lambda_1, \cdots, \lambda_i, \cdots, \lambda_n\} : \mathbf{H}\text{의 고유값}$$

이런 μ를 설정하기 위하여 Hangan[10,13]은 수식(2.12)와 같은 방법을 제안하였다. 여기서 μ의 변경은 역행렬과 오차함수의 재계산이 필요하다. 그러나 만약, μ를 수식(2.15)와 같이 H의 고유값 중 최소값($\lambda_{\min}$)의 절대값보다 크게 설정하면 수식(2.14)을 만족할 수 있다. 그러나 H의 차원이 커짐에 따라 고유값을 구하는 작업(특히, 고유방정식)은 많은 연산이 필요하다. 그러므로 행렬 H가 양의 정부호 행렬인지를 판단하고 최소 고유값을 근사화 하는 방법을 적용한다.

행렬 H가 양의 정부호 행렬인지를 판단하기 위해 본 논문에서는 Cholesky 분해를 이용한다. H가 수식(2.16)과 같이 분해되고, 대각 원소의 값이 모두 양이면 행렬 H는 양의 정부호 행렬이 된다. 즉 H의 고유값이 모두 양이 되어 근사 최소값을 가질 수 있다. 다음은 Cholesky 분해 알고리즘이며, pos.def는 positive definite를 의미한다.

$$\mathbf{H}=\mathbf{L}\mathbf{L}^T \quad \mathbf{L}: \text{lower triangular} \quad \cdots\cdots (2.16)$$

```
Cholesky Factorization
begin
  k := 0; pos.def:=true
  While pos.def and k < n
    k := k+1
    d := h_kk − Σ_{j=1}^{k−1} (l_kj)^2
    if d > 0
      l_kk := √d
      for i := k+1, ···, n
        l_ik := (h_ik − Σ_{j−1}^{k−1} l_ij l_kj) / l_kk
    else
      pos.def:=false
    end
  end
end
```

행렬 H의 차원이 커짐에 따라 수식(2.15)와 같은 최소 고유값을 찾는 과정은 매우 복잡하며 계산시간을 많이 요구한다. 따라서 본 논문에서는 μ를 근사적으로 구하기 위하여 주부분 행렬(principal submatrix)과 그 행렬의 다음과 같은 정리를 이용한다. k번째 주부분 행렬은 그림 2와 같이 n차 정방행렬 A의 처음 k개의 행과 열로 이루어진 A의 부분행렬이다.

정리 : 대칭행렬 H가 양의 정부호이기 위한 필요충분조건은 모든 주부분 행렬이 Cholesky 분해된다.

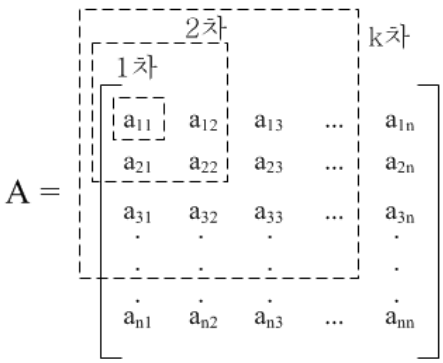


그림 2. k차 주부분 행렬
Fig 2. k-th principal submatrix

만약 k번째 주부분 행렬이 Cholesky 분해되면 μ를 변경하지 않지만, Cholesky 분해되지 않으면, 주부분 행렬의 최소 고유값으로 변경해야 한다. 그러나 이런 최소 고유값을 찾는 것도 k의 차수가 증가함에 따라 복잡한 계산이 될 수 있다. 따라서 행렬의 대각원소의 합($trace$)이 그 행렬의 고유값의 합을 나타내므로 주부분 행렬에 대한 $trace$의 절대값을 μ에 대한 근사값으로 계산한다.

이제 구체적인 알고리즘을 설명하자. 우선 학습 전에 μ을 임의의 작은 값(0.001)으로 초기화 한다. 그리고 오차함수를 계산한 후 행렬 H의 고유값이 모두 양의 값을 확인하기 위해 Cholesky 분해를 실행한다. 행렬 H의 Cholesky 분해 후, 양의 정부호 행렬이면 역행렬이 존재하여 가중치를 조정한 후에 μ를 수식(2.17)과 같이 $\beta(\beta>0, \beta=10)$로 축소하여 Gauss Newton 방법에 근사한 학습을 수행하도록 한다.

행렬 H가 Cholesky 분해가 되지 않으면 H의 주부분 행렬을 이용하여 다음과 같이 μ를 수정한다. 먼저 k차 주부분 행렬에서, $(k-1)$차까지의 주부분 행렬은 Cholesky 분해되고, k차 주부분 행렬에서 Cholesky 분해되지 않으면, 수식(2.18)과 같이 $\mathbf{H}_k^{sb}$의 $trace$의 절대값 저장한다. 그리고 $k+1$차 이상의 주부분 행렬의 계산을 위해서 $h_{kk} := h_{kk} + \mu_k$를 계산한다. μ_k를 $\mu_k := |trace|$로 계산한 것은 행렬의 $trace$가 고유치 합을 나타내는 성질을 이용한 것이다. 이와 같이 n차까지의 μ_k를 계산한 후 수식(2.19)와 같이 최대 μ_k을 선택하여 μ를 근사적으로 수정한다. μ를 수정하는 알고리즘은 다음과 같다.

μ Modification
If H:==pos.def
$\mu = \mu/\beta$ ..... (2.17)
else
for $k=1$ to $k<n$
$\mathbf{H}_k^{sb} := make\ k^{th}\ principal\ submatrix$
if $\mathbf{H}_k^{sb} :\neq$ pos.def
$\mu_k := \left| trace(\mathbf{H}_k^{sb}) \right|$ ..... (2.18)
$h_{kk} := h_{kk} + \mu_k \quad k=1,\cdots,k$
end
end
$\mu = \max(\mu_k)$ ..... (2.19)
end

Cholesky 분해와 μ수정 알고리즘을 이용한 전체적인 알고리즘은 다음과 같다.

1. μ를 임의의 작은 값으로 초기화한다.
2. 모든 입력/출력패턴을 입력하여 수식 (2.4)와 같이 오차를 계산한다.
3. 수식(2.7)과 (2.8)을 사용하여 Jacobian행렬을 계산한 후 행렬 H를 계산한다.
4. 행렬 H를 Cholesky 분해한다. 분해가 되면 수식(2.11)에 의해 가중치를 조정하고, μ를 수식(2.17)에 의해 축소한다.
5. 분해가 되지 않으면 k차 주부분 행렬을 구성하고 각 주부분 행렬에 대한 Cholesky 분해를 한 후, 최종적인 μ를 수식(2.19)와 같이 증가시킨다.
6. 수정된 μ를 이용하여 수식(2.11)과 같이 가중치를 조정한다.
7. 전체패턴에 대한 오차함수를 계산한 후, 정해진 오차범위까지 수렴하면 학습을 중단하고, 그렇지 않으면 2단계로 돌아간다.

제안된 방법은 μ를 일정하게 증가시키거나 감소시키는 기존 방법과 다르게 주부분 행렬을 이용하여 최소 고유값을 근사화하여 μ를 조정한다. 따라서 제안한 방법은 수식(2.12)에서와 같이 적당한 μ값을 얻기 위한 반복과정을 생략하며, 양의 정부호 행렬을 만들기 위한 근사한 μ값을 선택함으로써 보다 더 Gauss Newton 방법에 유사한 학습을 진행하여 학습속도를 개선할 수 있다.

III. 실험결과

LM학습에서 근사적인 μ값을 설정하여 학습속도를 개선하는 제안된 방법을 실험하기 위하여 본 논문에서는 두 가지 실험을 하였다. 우선 학습이 간단한 일반화된 XOR문제를 대상으로 제안한 방법을 확인한 후, 필기체 숫자인식[14]과 같은 실제적인 문제를 대상으로 실험하였다. 여기서 사용한 오차함수는 수식(2.20)과 같은 Mean Squared Error함수를 사용했다.

$$E = \frac{1}{QS}\sum_{q=1}^{Q}\sum_{j=1}^{S}(t_{qj} - a_{qj}^{K})^2 \quad \cdots\cdots (2.20)$$

LM학습의 구현은 전체적으로 Matlab의 Neural Network Toolbox을 이용하였고 제안된 방법을 적용하기 위하여 trainlm.m에서 수식(2.11)을 계산하는 부분을 수정하여 구현하였다. 일반화된 XOR문제에서는 MSE가 0.01이하로, 필기체 숫자인식 문제에서는 MSE가 0.02이하로 수렴할 때까지 학습했다.

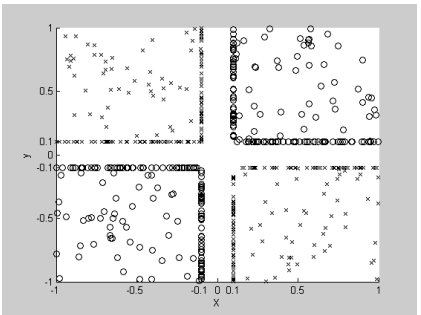


그림 3. 일반화된 XOR
Fig. 3. Generalized XOR

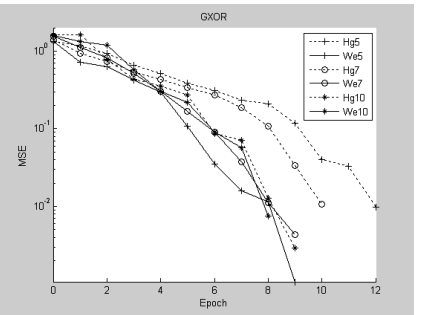


그림 4. 일반화된 XOR 결과
Fig. 4. Result of Generalized XOR

일반화된 XOR문제는 그림 3과 같이 학습패턴은 360개로 구성되며, 다층퍼셉트론의 노드수는 각각 입력층이 2개, 출력층 1개로 구성되며 은닉 노드수를 각각 5개, 7개, 10개로 변경하면서 학습하였다. 또한, Hangan의 방법과 비교하기 위하여 초기가중치 설정을 모두 동일하게 적용하였다.

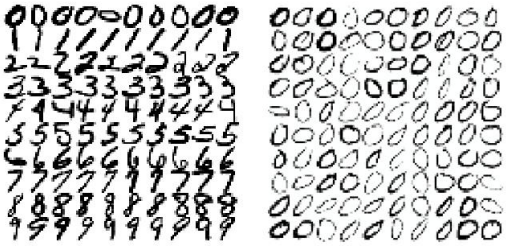
표 1과 그림 4는 일반화된 XOR문제의 학습결과를 나타낸다. 표 1의 단위는 epoch이고 그림 4의 Hg는 Hangan의 방법을 나타내며 We는 본 논문에서 제안한 방법이다. 표 1의 결과처럼 제안 방법은 기존 방법. 표3 ~1 epoch정도표 1일찍 수렴하여 학습속도의 개선을 보인다. 그림 4, 6, 7에서 그래프의 가독성을 높이기 위해 x축쉹결과형으로, y축쉹결대수적으로 증가시켰다.

표 1. 일반화된 XOR 결과
Table 1. Result of Generalized XOR

노드수 / 비교방법	5	7	10
Hangan 방법	12	10	9
제안한 방법	9	9	8

그림 4는 더 많은 정보를 나타내고 있는데, 은닉노드의 수가 같고 같은 epoch에서 제안 방법이 기존방법보다 MSE값을 더 많이 축소하여 최종오차에 먼저 수렴하는 결과를 나타낸다. 그러나 은닉노드의 수가 증가함에 따라 두 방법은 비슷한 결과를 보이는데, 이것은 은닉노드의 수가 증가함에 따라, 최종 가중치를 찾을 수 있는 오차함수의 공간이 증가하기 때문이다.

제안 알고리즘을 좀 더 복잡한 문제에 적용하기 위하여 그림 5와 같은 필기체 숫자인식 문제를 대상으로 실험하였다. 그림 5의 (a)는 학습패턴 중 각 숫자 10개씩을 나타낸 것이며, (b)는 숫자 0에 대한 학습패턴 100개를 모두 나타낸 것이다. 학습패턴은 각 숫자당 100개씩 총 1000개를 학습하였고, 다층퍼셉트론의 구성은 입력노드가 144개, 출력노드가 10개 그리고 은닉노드는 11개, 13개, 15개, 18개로 변경하면서 학습하였다. 여기서 다층퍼셉트론의 입력은 각 픽셀의 그레이값을 0 ~15로 구분한 실수값을 이용하였다.



(a) (b)

그림 5. 필기체 숫자 데이터
Fig 5. Handwritten Digit Data

필기체 숫자인식 문제는 LM알고리즘으로 학습하기에는 너무 많은 가중치를 가지고 있고, 초기 가중치의 설정에 따라 오차함수의 수렴이 크게 차이가 났다. 따라서 표 2는 초기가중치를 세 번씩 변경하면서 학습한 결과 중 가장 좋은 결과를 나타냈다. 그림 6는 은닉노드의 수가 11개, 13개인 경우이고 그림 7은 15개, 18개인 결과이다. 그림에서 DHg는 기존의 방법을 나타내며 DWe는 제안한 방법이다.

표 2. 필기체 숫자 결과
Table 2. Result in Handwritten Digit

노드수 / 비교방법	11	13	15	18
Hangan 방법	23	12	6	7
제안한 방법	14	9	5	6

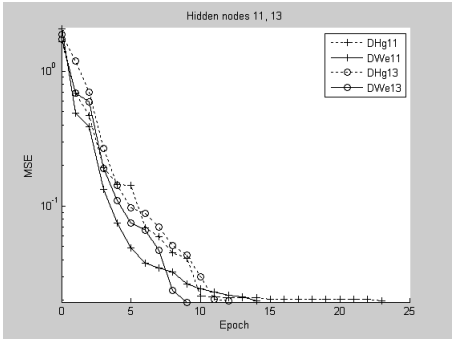


그림 6. 필기체 숫자 결과(11, 13노드)
Fig. 6. Result in Handwritten Digit(11, 13 nodes)

그림 6에서 제안한 방법은 조기포화 현상을 나타냈는데 이것은 출력층의 출력값이 시그모이드 함수의 목표값의 정반대 방향에 있기 때문에 발생한다. 이런 조기 포화현상은 출력값이 시그모이드 함수의 활성화 영역에 들어오면 조기포화 현상에서 벗어날 수 있었다.

그림 7의 결과는 그림 6의 경우보다 수렴속도가 늦은 결과가 발생했다. 이것은 은닉노드의 수를 증가한다고 해서 수렴속도가 빠른 것이 아니라, 오히려 많은 가중치는 불필요한 정보를 학습하는 과다학습의 결과를 보여주고 있다. 본 논문의 필기체 숫자인식 문제는 은닉노드의 수가 10 ~13개가 적정하였다.

표 2의 결과처럼 제안한 방법은 기존 방법보다 은닉노드의 수가 적은 경우, 더 빠른 학습속도를 보이고 있으며 은닉노드의 수가 증가할수록 비슷한 학습 속도를 나타내고 있다. 그러나 그림 6과 같이 같은 epoch이더라도 제안한 방법의 MSE가 더 작은 값을 가지고 있어 최종적으로 수렴하는 속도가 더 빠름을 알 수 있다.

실험 결과, 본 논문에서 제안한 방법이 μ값을 확대/축소하는 기존의 방법보다 학습속도의 향상을 가져왔다. 이것은 제안한 방법이 Jacobian 행렬의 주부분 행렬에 대한 고유값의 합을 이용하여 근사적인 μ값을 계산함으로 Jacobian 행렬의 역행렬을 바로 구할 수 있기 때문이다.

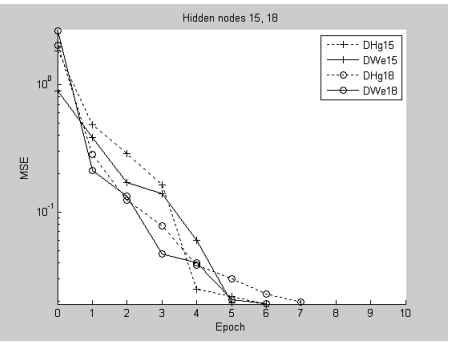


그림 7. 필기체 숫자 결과(15, 18노드)
Fig. 7. Result in Handwritten Digit(15, 18 nodes)

Ⅳ. 결론

Hangan이 제안한 LM학습은 오차함수에 대한 2차 도함수를 구하기 위하여 Jacobian 행렬을 이용한다. 그러나 이런 Jacobian 행렬이 가역행렬이 되기 위해 μ값을 확대시키거나 축소하는 과정이 필요하고 μ값의 변경에 따른 역행렬 계산이 추가적으로 필요하다.

따라서 본 논문에서는 μ값의 설정을 위해 단순한 확대/축소를 사용하는 것이 아니라, Jacobian 행렬의 주부분 행렬을 생성하고 주부분 행렬의 고유값 합을 이용하여 μ값을 설정함으로써 추가적인 역행렬 계산을 실행하지 않으므로 학습속도를 개선하는 방법을 제안하였다.

제안된 방법은 일반화된 XOR문제와 필기체 숫자인식 문제를 대상으로 실험하여 기존방법과 제안된 방법을 비교 분석하여 학습속도의 향상을 확인하였다.

참고문헌

〔1〕 D. E. Rumelhart and J. L. McClelland, *Parallel Distributed Processing*,MIT Press, Cambridge, MA, pp. 318-362, 1986.
〔2〕 김성완, "MLP 신경망을 위한 시공간 병렬처리모델," 한국컴퓨터정보학회 논문지, 제10권 제5호, 95-102쪽, 2005년 11월
〔3〕 김광백, 조재현, "퍼지 신경망을 이용한 자동차 번호판 인식 시스템," 한국컴퓨터정보학회 논문지, 제12권 제5호, 313-319쪽, 2007년 11월
〔4〕 John Moody and Christian J. Darken, "Fast

Learning in Networks of Locally-Tuned Processing Units," Neural Computation, vol. 1, pp. 281-294, 1989.
〔5〕 T. P. Vogal, J. K. Mangis, A. K. Zigler, W. T. Zink and D. L. Alkon, "Accelerating the convergence of the backpropagation method," Biological Cybernetics, vol. 59, pp. 256-264, Sept. 1988.
〔6〕 T. Tollenaere, "SuperSAB: Fast adaptive back propagation with good scaling properties," Neural Networks, vol. 3, no. 5, pp. 561-573, 1990.
〔7〕 M. Kordos and W. Duch, "Variable step search algorithm for feedforward networks," Neurocomputing, vol. 71, pp. 2470-2480, 2008.
〔8〕 C. Charalambous, "Conjugate gradient algorithm for efficient training of artificial neural networks," IEEE Proceedings, vol. 139, no. 3, pp. 301-310, 1992.
〔9〕 Rolnald E. Miller, *Optimization*, John Wiley & Son, INC. pp. 358-362, 2000.
〔10〕 M. T. Hagan and M. Menhaj, "Training feedforward networks with the Marquardt algorithm," IEEE Transactions on Neural Networks, vol. 5, no. 6, pp. 989-993, 1994.
〔11〕 M. A. Costa, A. P. Braga and B. R. Menezes, "Improving generation of MLPs with sliding mode control and the Levenberg-Marquardt algorithm," Neurocomputing, vol. 70, pp. 1342-1347, 2007.
〔12〕 L. E. Scales, *Introduction to Non-Linear Optimization*, New York: Springer-Verlag, 1985.
〔13〕 M. T. Hagan, H. B. Demuth, M. Beale, *Neural Network Design*, PWS Publishing Company, 1995.
〔14〕 J. J. Hull, "A database for handwritten text recognition research," IEEE Transaction Pattern and Machine Intell., vol. 16, pp. 550-554, 1994.

저 자 소 개

곽영태

1993년 충남대학교 컴퓨터공학과 학사
1995년 충남대학교 컴퓨터공학과 석사
2001년 충남대학교 컴퓨터공학과 박사
2001년 ~ 2002년
한국전자통신연구원 선임연구원
2002년 3월 ~ 2008년 2월
익산대학 컴퓨터과학과 조교수
2008년 3월 ~ 현재
전북대학교 응용시스템공학부 부교수
관심분야 : 신경회로망, 패턴인식, 영상처리

신정훈

1982년 숭실대학교 전자계산학과 학사
1991년 충북대학교 전산통계학과 석사
1999년 충북대학교 전자계산학과 박사
1992년 3월 ~ 2008년 2월
익산대학 컴퓨터과학과 교수
2008년 3월 ~ 현재
전북대학교 응용시스템공학부 교수
관심분야 : 멀티미디어 DBMS, 임베디드시스템