Neural Networks for Machine Learning

The Backpropagation Learning Procedure

산업경영공학과 201533258 장예훈

1 역전파 알고리즘(Backpropagation Algorithm)이란?

- 최적화 방법과 함께 인공신경망을 학습시킬 때 많이 이용
- Algorithm

i 신경망에 훈련 샘플을 부여

ii Actual Output과 Target Output을 비교

iii 각 출력 뉴런에서의 에러를 계산

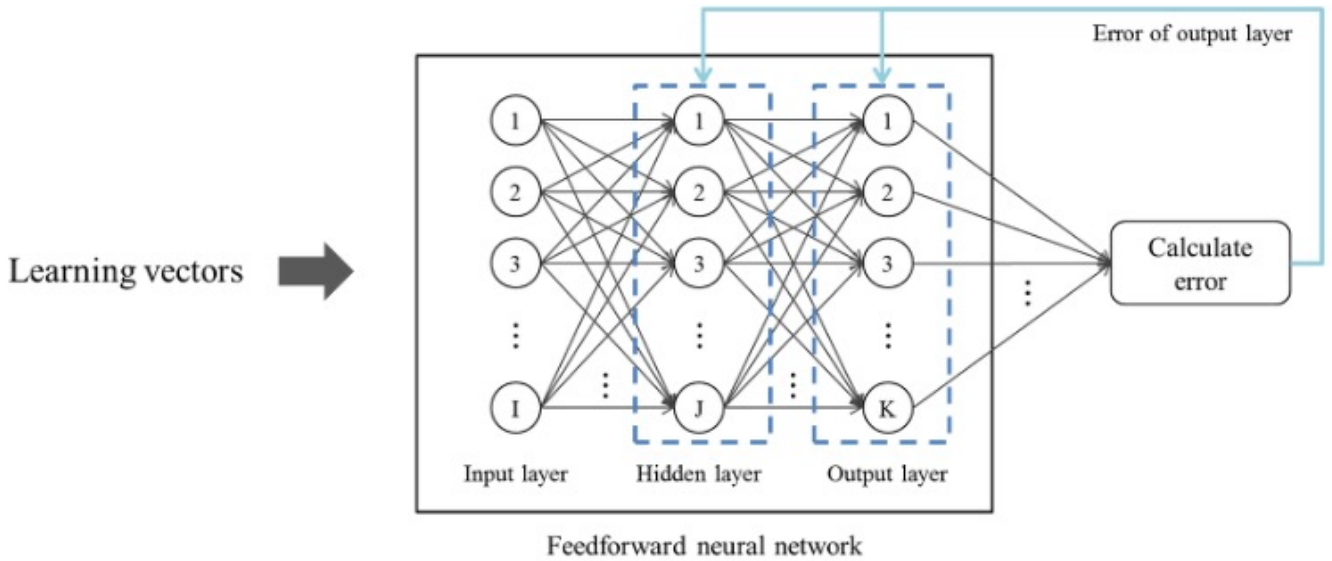
iv 각 뉴런에 대해 에러, 실제 출력, Scaling Factor 를 계산 후 Target Value와의 오차 계산 → Local Error

v 각 뉴런으로 들어오는 Connection에 대해 가중치를 사용, 이전 레벨의 뉴런에 대해 Local Error를 위한 Blame을 할당

vi 이전 레벨의 뉴런에 대해 위의 단계를 반복

vii 이때 각 뉴런에 대해 Error 로서 Blame 을 사용

1 역전파 알고리즘(Backpropagation Algorithm)이란?



- 다중 퍼셉트론 (Multi-Layer Perceptron)에서는 은닉층(Hidden Layer)의 목적값을 결정 할 수 없기 때문에 학습수행 불가
- 역전파 알고리즘은 출력층(Output Layer)의 오차를 역전파하여 은닉층을 학습
  → 다중 퍼셉트론의 문제 해결
- 다양한 종류의 Feedforward Neural Netaworks(FNNs)의 학습알고리즘으로도 사용

2 역전파 알고리즘의 단계

1 전파(propagation) 단계 : 학습벡터로부터 실제 출력값 출력

→ 목적값(target value)과 실제 출력값의 오차를 각 Hidden Layer에 전파

2 가중치 수정 단계 : 전파된 오차를 이용하여 가중치를 수정

3 역전파 알고리즘과 Gradient Descent Rule

- 역전파는 변경가능한 가중치(Weight)에 대해 에러의 기울기를 계산하는데 사용

$$v_{kj}(t+1) = v_{kj}(t) + \eta\delta_{nk}z_{nj}$$

  * η는 가중치가 특정한 점으로 이동할 때 한번의 시행에서 얼만큼 이동할 것인지를 나타내며, 학습률(Learning Rate)라고 함

- 기울기는 에러를 최소화하는 가중치를 찾기 위한 경사하강법에서 사용
- 역전파 알고리즘은 Gradient Descent Rule과 결합되어 인공신경망의 학습알고리즘으로 많이 사용

4 역전파 알고리즘과 Chain Rule

- 역전파 학습 알고리즘은 최소평균자승 알고리즘 (Least Mean Square Algorithm)의 비선형적 확장
- 미분의 반복규칙(Chain Rule)을 여러 번 적용하여 확률 근사치 프레임워크유도

  ex) A라는 사람이 멀리 떨어져 있는 E라는 정보를 전달하기 위해 B, C, D 중간 사람을 거치는 것

5 역전파 알고리즘과 Delta Rule

- 앞의 방식의 역전파는 3층 이상의 깊은 뉴럴넷에서 수식을 계산하기 복잡
- Delta Rule 적용

  : Error를 Weighted Sum으로 미분한 것을 '델타'로 정의

$$\triangle w_{ij} = \epsilon(t_j - y_j)y_i(1 - y_i)x_i$$

6 역전파 구현식

2) W_{ji}의 $gradient$를 계산 하는 방법.

$$E=\frac{1}{2}\sum_{k}\{y_k-t_K\}^2$$

즉, k개의 $output\ node$에서 발생한 에러의 종합을 구한다.
이때 에러를 각 파라미터로 편미분하여 각 파라미터가 학습할 기울기를 구한다.

w_{ji} : $weight\ from\ i\ node\ to\ j\ node$

$$\nabla E_{ji} = \frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} = \delta_j z_i$$

(δ_j : $j\ node's\ error$, z_i : $i\ node's\ activation$)

$$\delta_j \equiv \frac{\partial E}{\partial a_j}\ ,\ a_j=\sum_i w_{ji}z_i\ \ then$$

$$\frac{\partial a_j}{\partial w_{ji}} = \frac{\partial \sum_i \mathbf{w_{ji}z_i}}{\partial \mathbf{w_{ji}}} = \mathbf{z_i}$$

6 역전파 구현식

즉, 어떤 *weight*의 *gradient*($\frac{\partial E}{\partial w_{ji}}$)는 *j*노드의 *error*와 *i*노드의 *activation value*를 곱한다.

다시 말하면, *weight*가 가리키고 있는 *node*의 *error*와 그 *weight*가 출발한 *node*의 *activation*값을 곱하는 것이다. 이는 위쪽 *node*의 *error*에 대해, 아래쪽 *node*가 *activation*한, 즉 기여한 만큼을 *weight*가 학습하겠다는 것이다.

6 역전파 구현식

1) 아래는 *j node*의 에러를 *error back-propagation*을 통해 계산

δ_j는 *j*번째 노드로 전파 된 *node error*

$$\delta_j = \frac{\partial E}{\partial a_j} = \sum_k \frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial a_j}\ ,\ \delta_k \equiv \frac{\partial E}{\partial a_k}\ ,\ a_k = \sum_j w_{kj} z_j\ ,\ z_j = h(a_j)\quad \textbf{, then}$$

$$\sum_k \frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial a_j} = \sum_k \delta_k \frac{\partial a_k}{\partial a_j} = \sum_k \delta_k \frac{\partial \Sigma_j\{w_{kj} h(a_j)\}}{\partial a_j} = \sum_k \delta_k\ h'(a_j) w_{kj} = h'(a_j)\sum_k w_{kj}\,\delta_k$$

$$\textit{therefore},\ \delta_j = h'(a_j)\sum_k w_{kj}\,\delta_k$$

즉 *j* 노드의 에러는 *k-layer*의 노드들에 대한 *weighted sum of error*에
h'(j 노드의 *value)*를 곱한 것이다.

이는 우선 *k-layer* 노드들의 에러를 *weighted sum*으로 모은다음,
$h'(a_j)$으로 그 노드의 *value*에 대한 *h* 함수의 기울기를 곱한것이다.

$$\nabla E_{ji} = \frac{\partial E}{\partial w_{ji}} = \delta_j x_j$$

(노드의 에러*노드의 *input value*로 *weight*의 *gradient*를 계산한다.)

THANK YOU

감사합니다!