## Error Backpropagation of Feedforward Networks

참고. Learning Internal Representations by Error Propagation
in the book “Parallel Distributed Processing” 1986
by Rumelhart, Hinton, Williams

### 1. 그래디언트.

오류 역전파는 사실 합성 함수에서 그래디언트를 계산하는 것과 같다. 따라서 합성 함수의 그래디언트를 이해한다면 오류 역전파에 대한 모든 것을 이해하게 된다.

다음 합성 함수를 생각해 보자.

$$\mathbb{R}^p \xrightarrow{F} \mathbb{R}^q \xrightarrow{g} \mathbb{R}$$

변수 $\mathbf{x} = (x_1, \ldots, x_p)$, $\mathbf{y} = (y_1, \ldots, y_q)$를 대입하여 적으면 다음과 같다.

$$\mathbf{y} = F(\mathbf{x}),\ z = g(\mathbf{y})$$

함수 $F$는 $q$ 개의 함수 $y_1 = f_1(\mathbf{x}), \ldots, y_q = f_q(\mathbf{x})$로 이루어져 있으며

$$F(\mathbf{x}) = (f_1(\mathbf{x}),\ \ldots, f_q(\mathbf{x}))$$

또는

$$F(\mathbf{x}) = [f_1(\mathbf{x})\ \ldots\ f_q(\mathbf{x})]^T$$

로 적을 수 있다.

두 함수 $z = g(\mathbf{y})$와 $z = (g \circ F)(\mathbf{x})$의 그래디언트는 각각

$$\nabla g(\mathbf{y}) = \begin{bmatrix} \dfrac{\partial z}{\partial y_1} \\ \vdots \\ \dfrac{\partial z}{\partial y_q} \end{bmatrix},\ \nabla (g \circ F)(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial z}{\partial x_1} \\ \vdots \\ \dfrac{\partial z}{\partial x_p} \end{bmatrix}$$

이다. 합성 함수의 미분법은 $\nabla (g \circ F)(\mathbf{x})$가

$$\begin{bmatrix} \dfrac{\partial z}{\partial x_1} \\ \vdots \\ \dfrac{\partial z}{\partial x_p} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial y_1}{\partial x_1} & \cdots & \dfrac{\partial y_q}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial y_1}{\partial x_p} & \cdots & \dfrac{\partial y_q}{\partial x_p} \end{bmatrix} \begin{bmatrix} \dfrac{\partial z}{\partial y_1} \\ \vdots \\ \dfrac{\partial z}{\partial y_q} \end{bmatrix} \tag{1.1}$$

임을 말해준다. $F$의 야코비 행렬 $J(F)$를 써서 표현하면

$$\nabla (g \circ F)(\mathbf{x}) = J(F)(\mathbf{x})^T \nabla g(\mathbf{y})$$

이다. 간단히

$$\nabla (g \circ F) = J(F)^T \nabla g \tag{1.2}$$

로 나타내기도 한다.

일반적으로

$$\mathbb{R}^{p_1} \xrightarrow{F_1} \cdots \xrightarrow{F_{n-1}} \mathbb{R}^{p_n} \xrightarrow{F_n} \mathbb{R}^q \xrightarrow{g} \mathbb{R} \tag{1.3}$$

에서 합성 함수 $g \circ F_n \circ \cdots \circ F_1$의 그래디언트는

$$\nabla (g \circ F_n \circ \cdots \circ F_1) = J(F_1)^T \cdots J(F_n)^T \nabla g$$

이다. $\mathbb{R}^q$에서 정의된 $g$의 그래디언트 $\nabla g$를 한 단계씩 차례로 이동(역전파)시켜 각 단계의

그래디언트를 얻을 수 있다.

$$\nabla(g \circ F_n \circ \cdots \circ F_1) \xleftarrow{J(F_1)^T} \cdots \xleftarrow{J(F_{n-1})^T} \nabla(g \circ F_n) \xleftarrow{J(F_n)^T} \nabla g \tag{1.4}$$

이 과정은 네트워크에서 오류 역전파의 그것과 일치한다.

## 2. Feedforward networks.

Internal layer가 하나인 경우 오류 역전파를 살펴보자. Internal layer가 없거나 둘 이상인 경우는 하나인 경우와 유사하다.

### 2.1 네트워크

Internal layer가 하나이고 bias가 없는 네트워크는 다음과 같다.

$$\begin{bmatrix} x_1^0 \\ \vdots \\ x_p^0 \end{bmatrix} \xrightarrow{W^1} \begin{bmatrix} z_1^1 \\ \vdots \\ z_q^1 \end{bmatrix} \xrightarrow{F^1} \begin{bmatrix} x_1^1 \\ \vdots \\ x_q^1 \end{bmatrix} \xrightarrow{W^2} \begin{bmatrix} z_1^2 \\ \vdots \\ z_r^2 \end{bmatrix} \xrightarrow{F^2} \begin{bmatrix} x_1^2 \\ \vdots \\ x_r^2 \end{bmatrix} \tag{2.1}$$

$x_i^0$는 input, $x_i^2$는 output, $W^k$는 가중치 행렬, $F^k$는 활성 함수이다. 이 네트워크에 손실 함수 $g$를 붙여 적으면

$$\mathbb{R}^p \xrightarrow{W^1} \mathbb{R}^q \xrightarrow{F^1} \mathbb{R}^q \xrightarrow{W^2} \mathbb{R}^r \xrightarrow{F^2} \mathbb{R}^r \xrightarrow{g} \mathbb{R} \tag{2.2}$$

가 된다.

기호를 단순화하기 위하여

$$\mathbf{x}^0 = [x_1^0 \cdots x_p^0]^T$$
$$\mathbf{z}^1 = [z_1^1 \cdots z_q^1]^T$$

등으로 적기로 한다. 그러면

$$\mathbf{z}^1 = W^1 \mathbf{x}^0$$
$$\mathbf{x}^1 = F^1(\mathbf{z}^1)$$

이 된다. target은

$$\mathrm{t} = [t_1 \cdots t_r]^T$$

로 적는다. 손실 함수의 값, 곧 오류는

$$E = g(\mathbf{x}^2)$$

로 나타낸다.

가중치 행렬의 성분은

$$W^1 = \begin{bmatrix} w_{11}^1 & \cdots & w_{1p}^1 \\ \vdots & \ddots & \vdots \\ w_{q1}^1 & \cdots & w_{qp}^1 \end{bmatrix}$$

로 적는다.

### 2.2 그래디언트

(2.2)의 각 단계에서의 그래디언트를 (1.4)와 같이 구한다.

$$\nabla(g \circ F^2 \circ W^2 \circ F^1) \xleftarrow{J(F^1)^T} \nabla(g \circ F^2 \circ W^2) \xleftarrow{J(W^2)^T} \nabla(g \circ F^2) \xleftarrow{J(F^2)^T} \nabla g \tag{2.3}$$

이것이 오류 역전파를 계산하기 쉽게 해준다. 각 단계의 그래디언트를 바로 전 단계의 그래디언트로부터 구할 수 있기 때문이다.

그런데 여기서 생각해 볼 것은 그래디언트를 구할 때에는 네트워크의 독립 변수가 $W^1$과 $W^2$라는 것이다. (2.3)에서는 $W^1$과 $W^2$가 함수이기 때문에 그것을 직접 적용할 수 없다.

먼저 $E$를 $W^2$로 미분하기 위하여 함수 $\phi^2$를

$$\phi^2(W^2) = W^2\mathbf{x}^1$$

이라 정의하자. 오류는

$$E = (g \circ F^2 \circ \phi^2)(W^2) \tag{2.4}$$

가 된다. 합성 함수로 표현하면

$$\mathbb{R}^{qr} \xrightarrow{\phi^2} \mathbb{R}^r \xrightarrow{F^2} \mathbb{R}^r \xrightarrow{g} \mathbb{R}^r$$

이다. $W^2$는 $r \times q$ 행렬이므로 $\mathbb{R}^{qr}$의 원소로 간주할 수 있다.

결국 (2.4)의 그래디언트 $\nabla(g \circ F^2 \circ \phi^2)$를 구하는 것이 목표가 된다. 이것을 **$W^2$에 대한 $E$의 그래디언트**라 하고 $\nabla_{W^2}E$로 쓰기로 하자.

$$\nabla_{W^2}E = \nabla(g \circ F^2 \circ \phi^2)(W^2)$$

이것을 계산해 보자. $z_k^2 = \sum_{l=1}^{q} w_{kl}^2 x_l^1$에서

$$\begin{aligned} \frac{\partial E}{\partial w_{ij}^2} &= \frac{\partial E}{\partial z_i^2}\frac{\partial z_i^2}{\partial w_{ij}^2} \\ &= \frac{\partial E}{\partial z_i^2} x_j^1 \end{aligned}$$

를 얻을 수 있다. 이 식을 행렬로 나타내면

$$\begin{bmatrix} \frac{\partial E}{\partial w_{11}^2} & \cdots & \frac{\partial E}{\partial w_{1q}^2} \\ \vdots & \ddots & \vdots \\ \frac{\partial E}{\partial w_{r1}^2} & \cdots & \frac{\partial E}{\partial w_{rq}^2} \end{bmatrix} = \begin{bmatrix} \frac{\partial E}{\partial z_1^2} \\ \vdots \\ \frac{\partial E}{\partial z_r^2} \end{bmatrix} \begin{bmatrix} x_1^1 \cdots x_q^1 \end{bmatrix} = \nabla(g \circ F^2)\,\mathbf{x}^{1T}$$

가 된다. 야코비 행렬과 $\mathbf{x}^1$으로 표현하면

$$\nabla_{W^2}E = J(F^2)^T \nabla g\, \mathbf{x}^{1T} \tag{2.5}$$

이다.

◇ $W^2$를 벡터로 표현하여 계산할 때는

$$\nabla_{W^2}E = J(\phi^2)J(F^2)^T \nabla g$$

가 된다. 식 (2.5)는 $W^2$를 행렬로 표현했을 때의 식이다.

같은 방법으로 $W^1$에 대한 $E$의 그래디언트를 구한다.

$$\mathbb{R}^{qp} \xrightarrow{\phi^1} \mathbb{R}^q \xrightarrow{F^1} \mathbb{R}^q \xrightarrow{W^2} \mathbb{R}^r \xrightarrow{F^2} \mathbb{R}^r \xrightarrow{g} \mathbb{R}$$

이 때에는 $W^2$가 함수로 작용하므로

$$\nabla_{W^1}E = J(F^1)^T W^{2T} J(F^2)^T \nabla g\, \mathbf{x}^{0T} \tag{2.6}$$

를 얻는다. $W^2$가 행렬이므로 $J(W^2) = W^2$이다.

training은 $W^k$를 $W^k - \eta \nabla_{W^k} E$로 반복하여 바꾸는 것으로 이루어진다. 이렇게 행렬로 계산하기 위하여 $\nabla_{W^k} E$를 행렬로 표현하였다. $\eta$는 learning rate이다.

## 2.3 bias

네트워크 (2.1), (2.2)에 bias를 추가하면

$$\begin{bmatrix} x_1^0 \\ \vdots \\ x_p^0 \end{bmatrix} \xrightarrow{(b^1, W^1)} \begin{bmatrix} z_1^1 \\ \vdots \\ z_q^1 \end{bmatrix} \xrightarrow{F^1} \begin{bmatrix} x_1^1 \\ \vdots \\ x_q^1 \end{bmatrix} \xrightarrow{(b^2, W^2)} \begin{bmatrix} z_1^2 \\ \vdots \\ z_r^2 \end{bmatrix} \xrightarrow{F^2} \begin{bmatrix} x_1^2 \\ \vdots \\ x_r^2 \end{bmatrix} \tag{2.7}$$

$$\mathbb{R}^p \xrightarrow{(b^1, W^1)} \mathbb{R}^q \xrightarrow{F^1} \mathbb{R}^q \xrightarrow{(b^2, W^2)} \mathbb{R}^r \xrightarrow{F^2} \mathbb{R}^r \xrightarrow{g} \mathbb{R} \tag{2.8}$$

이 된다. bias는

$$b^1 = \begin{bmatrix} b_1^1 \\ \vdots \\ b_q^1 \end{bmatrix}, \ b^2 = \begin{bmatrix} b_1^2 \\ \vdots \\ b_r^2 \end{bmatrix}$$

이고, 두 함수 $(b^1, W^1)$과 $(b^2, W^2)$는 각각

$$(b^1, W^1)(\mathbf{x}^0) = b^1 + W^1 \mathbf{x}^0$$
$$(b^2, W^2)(\mathbf{x}^1) = b^2 + W^2 \mathbf{x}^1$$

로 정의된다.

bias가 있는 경우 그래디언트를 계산해보자. $W^k$에 대한 $E$의 그래디언트는 식 (2.5), (2.6)과 일치한다. **$b^k$에 대한 $E$의 그래디언트** $\nabla_{b^k} E$를 구해보자.

먼저 $\nabla_{b^2} E$를 구한다. $z_i^2 = b_i^2 + \sum_{j=1}^{q} w_{ij}^2 x_j^1$이므로

$$\begin{aligned} \frac{\partial E}{\partial b_i^2} &= \frac{\partial E}{\partial z_i^2} \frac{\partial z_i^2}{\partial b_i^2} \\ &= \frac{\partial E}{\partial z_i^2} \end{aligned}$$

임을 알 수 있다. 따라서

$$\begin{bmatrix} \dfrac{\partial E}{\partial b_1^2} \\ \vdots \\ \dfrac{\partial E}{\partial b_r^2} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial E}{\partial z_1^2} \\ \vdots \\ \dfrac{\partial E}{\partial z_r^2} \end{bmatrix}$$

이고 야코비 행렬을 써서

$$\nabla_{b^2} E = J(F^2)^T \nabla g \tag{2.9}$$

를 얻는다. 마찬가지 방법으로

$$\nabla_{b^1} E = J(F^1)^T W^{2T} J(F^2)^T \nabla g \tag{2.10}$$

를 얻을 수 있다. 함수 $(b^2, W^2)$의 야코비 행렬은 bias가 없을 때와 일치한다. 곧

$$J(b^2, W^2) = J(W^2) = W^2$$

이다.

## 3. 활성 함수와 손실 함수

training에서는 네 식 (2.5), (2.6), (2.9), (2.10)을 계산한다. 이 식들은 활성 함수와 손실 함수에 따라 간단하게 줄일 수도 있다.

### 3.1 활성 함수

활성 함수 $F^1$과 $F^2$는 각 성분별로 정의되고 각 성분의 함수가 같은 경우가 대부분이다. 곧,

$$\begin{aligned} F^1(\mathbf{z}^1) &= (f^1(z_1^1),\ \ldots, f^1(z_q^1)) \\ F^2(\mathbf{z}^2) &= (f^2(z_1^2),\ \ldots, f^2(z_r^2)) \end{aligned} \tag{3.1}$$

와 같은 형태로 정의된다. 이것은 빠른 계산을 위한 현실적인 선택이다. 이론적으로는 야코비 행렬을 구할 수 있는, 미분 가능한 함수이면 충분하다.

식 (3.1)과 같이 정의된 두 활성 함수 $F^1$과 $F^2$의 야코비 행렬은 대각행렬이다. 예를 들어 $F^2$의 야코비 행렬은 다음과 같다.

$$J(F^2) = \begin{bmatrix} f^{2\prime}(z_1^2) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & f^{2\prime}(z_r^2) \end{bmatrix}$$

대각행렬과 벡터의 곱은 Hadamard product로 고쳐 쓸 수 있으므로 $\nabla(g \circ F^2)$는

$$J(F^2)^T \nabla g = \begin{bmatrix} f^{2\prime}(z_1^2) \\ \vdots \\ f^{2\prime}(z_r^2) \end{bmatrix} * \nabla g \tag{3.2}$$

로 계산할 수 있다. *는 Hadamard product이다.

◇ 식 (3.2)는 (3.1)과 같이 각 성분별로 정의된 함수에만 적용할 수 있다. 일반적인 활성 함수, 예를 들어 softmax에는 이 식을 적용할 수 없다.

**3.1.1 logistic sigmoid.** logistic sigmoid는 $\sigma(x) = \dfrac{1}{1+e^{-x}}$로 정의된다. 이 함수의 도함수는

$$\sigma'(x) = \sigma(x)(1-\sigma(x))$$

를 만족한다. 따라서 $f^k$가 logistic sigmoid이면

$$f^{k\prime}(z_i^k) = x_i^k(1-x_i^k) \tag{3.3}$$

가 성립한다.

**3.1.2 hyperbolic tangent.** hyperbolic tangent는 $\tanh(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$로 정의된다. 이 함수의 도함수는

$$\tanh'(x) = 1 - \tanh(x)^2$$

를 만족한다. 따라서 $f^k$가 hyperbolic tangent이면

$$f^{k\prime}(z_i^k) = 1 - (x_i^k)^2$$

이 성립한다.

**3.1.3 softmax.** 함수 $F^2$가 softmax이면

$$F^2(z_1^2, \ldots, z_r^2) = \left( \frac{e^{z_1^2}}{e^{z_1^2} + \cdots + e^{z_r^2}}, \ldots, \frac{e^{z_r^2}}{e^{z_1^2} + \cdots + e^{z_r^2}} \right) \tag{3.4}$$

이다. 각 성분

$$x_i^2 = \frac{e^{z_i^2}}{e^{z_1^2} + \cdots + e^{z_r^2}}, \ k = 1, \ldots, r$$

를 미분하여 야코비 행렬

$$J(F^2) = \begin{bmatrix} x_1^2(1 - x_1^2) & \cdots & -x_1^2 x_r^2 \\ \vdots & \ddots & \vdots \\ -x_r^2 x_1^2 & \cdots & x_r^2(1 - x_r^2) \end{bmatrix} \tag{3.5}$$

를 얻을 수 있다.

### 3.2 손실 함수

이 보고서에서 $g$로 나타낸 손실 함수는 mean squared error와 cross entropy가 있다. 전자는 유클리드 거리를 제곱한 것이다. 후자는 $r = 1$인 경우 binary cross entropy라 불리는 형태로 사용되며, $r > 1$인 경우 categorical cross entropy라 불리는 형태가 사용된다.

#### 3.2.1 mean squared error

Mean squared error는

$$g(\mathrm{x}^2) = \frac{1}{2} \|\mathrm{t} - \mathrm{x}^2\|^2 = \frac{1}{2} \sum_{i=1}^{r} (t_i - x_i^2)^2$$

로 정의된다. 계산을 줄이기 위하여 $\frac{1}{2}$을 곱하였다. 이 함수의 그래디언트는

$$\nabla g(\mathrm{x}^2) = \mathrm{x}^2 - \mathrm{t} = \begin{bmatrix} x_1^2 - t_1 \\ \vdots \\ x_r^2 - t_r \end{bmatrix}$$

이다.

**3.2.2 binary cross entropy와 logistic sigmoid.** 각 target $t_i$가 베르누이 분포를 따르는 확률이거나 그렇게 이해되는 경우 손실 함수로 binary cross entropy를 사용할 수 있다.

$$g(\mathrm{x}^2) = -\sum_{i=1}^{r} (t_i \log x_i^2 + (1 - t_i) \log (1 - x_i^2))$$

이 때 $g$의 그래디언트는

$$\nabla g(\mathrm{x}^2) = \begin{bmatrix} \dfrac{x_1^2 - t_1}{x_1^2(1 - x_1^2)} \\ \vdots \\ \dfrac{x_r^2 - t_r}{x_r^2(1 - x_r^2)} \end{bmatrix} \tag{3.6}$$

이다.

이 형태에서 logistic sigmoid를 활성함수로 사용하면 그래디언트의 식이 매우 단순하다. $F^2$가 식 (3.1)과 같이 주어지고 각 성분 $f^2$가 logistic sigmoid이면, 식 (3.2), (3.3), (3.6)으로부터

$$\nabla(g \circ F^2)(\mathrm{z}^2) = \mathrm{x}^2 - \mathrm{t} \tag{3.7}$$

를 얻을 수 있다.

**3.2.3 categorical cross entropy와 softmax.** target $t_1, \ldots, t_r$가 $t_i \ge 0$, $\sum t_i = 1$을 만족할 경우, 곧 확률인 경우에는 손실 함수 $g$를

$$g(\mathrm{x}^2) = -\sum_{k=1}^{c} t_k \log x_k^2$$

로 정의할 수 있다. 이것을 categorical cross entropy라 한다. 이 때 $g$의 그래디언트는

$$\nabla g(\mathrm{x}^2) = -\begin{bmatrix} t_1/x_1^2 \\ \vdots \\ t_r/x_r^2 \end{bmatrix} \tag{3.8}$$

이다. 이 형태에서는 softmax를 활성 함수로 사용하면 그래디언트를 쉽게 계산할 수 있다. $F^2$가 (3.4)와 같이 주어지면, 식 (3.5), (3.8)로부터

$$\nabla(g \circ F^2)(\mathrm{z}^2) = \mathrm{x}^2 - \mathrm{t} \tag{3.9}$$

를 얻을 수 있다. 식 (3.7)과 동일한 결과이다.

## 4. 프로그래밍

Feedforward network는 많은 양의 데이터를 다룬다. 따라서 벡터에 대하여 정의된 식 (2.5), (2.6), (2.9), (2.10)은 그대로 적용되지 않는다. 코딩에 사용할 수 있는 표현법을 알아보자.

### 4.1 열 벡터 표현

$p$ 차원 input $x_1^0, \ldots, x_n^0$과 대응되는 $r$ 차원 target $t_1, \ldots, t_n$이 주어졌다고 하자. 이 데이터를 식 (2.7), (2.8)의 네트워크에 적용해 보자.

먼저 input과 target을 결합하여 두 행렬

$$X^0 = \begin{bmatrix} x_{11}^0 \cdots x_{1n}^0 \\ \vdots \ \ddots \ \vdots \\ x_{p1}^0 \cdots x_{pn}^0 \end{bmatrix}, \quad T = \begin{bmatrix} t_{11} \cdots t_{1n} \\ \vdots \ \ddots \ \vdots \\ t_{r1} \cdots t_{rn} \end{bmatrix}$$

을 정의한다. 마찬가지 방법으로

$$Z^1 = \begin{bmatrix} z_{11}^1 \cdots z_{1n}^1 \\ \vdots \ \ddots \ \vdots \\ z_{q1}^1 \cdots z_{qn}^1 \end{bmatrix}, \quad X^1 = \begin{bmatrix} x_{11}^1 \cdots x_{1n}^1 \\ \vdots \ \ddots \ \vdots \\ x_{q1}^1 \cdots x_{qn}^1 \end{bmatrix}$$

$Z^2$, $X^2$ 등을 정의한다. 이들은 다음 식을 만족한다.

$$Z^1 = b^1 + W^1 X^0, \quad X^1 = F^1(Z^1), \ \ldots$$

$X^2$에 대한 $g$의 그래디언트를 $r \times n$ 행렬로 다음과 같이 나타낸다.

$$\nabla g = \nabla_{X^2} E = \begin{bmatrix} \frac{\partial E}{\partial x_{11}^2} \cdots \frac{\partial E}{\partial x_{1n}^2} \\ \vdots \quad \ddots \quad \vdots \\ \frac{\partial E}{\partial x_{r1}^2} \cdots \frac{\partial E}{\partial x_{rn}^2} \end{bmatrix}$$

그래디언트는 일반적으로 벡터로 표현되지만 여기서는 계산을 위하여 행렬로 나타낸다. $\nabla_{Z^2} E$, $\nabla_{X^1} E$, $\nabla_{Z^1} E$ 등도 유사하게 행렬로 나타내기로 한다. $F^1(Z^1)$은 행렬

$$F^1(Z^1) = \begin{bmatrix} f^1(z_{11}^1) \cdots f^1(z_{1n}^1) \\ \vdots \quad \ddots \quad \vdots \\ f^1(z_{q1}^1) \cdots f^1(z_{qn}^1) \end{bmatrix}$$

이며 $F^2(Z^2)$도 유사하다. 이 행렬의 각 성분의 도함수를 다시 행렬로 표현한다.

$$F^{1\prime}(Z^1) = \begin{bmatrix} f^{1\prime}(z_{11}^1) \cdots f^{1\prime}(z_{1n}^1) \\ \vdots \quad \ddots \quad \vdots \\ f^{1\prime}(z_{q1}^1) \cdots f^{1\prime}(z_{qn}^1) \end{bmatrix}$$

이 행렬은 $F^1$의 야코비 행렬과 다름을 확인하기 바란다. 참고로 야코비 행렬 $J(F^1)$은 $qn \times qn$ 행렬이다. $F^{2\prime}(Z^2)$도 마찬가지로 $r \times n$ 행렬로 나타낸다.

주의를 기울여 계산하면 행렬 $\nabla_{Z^2} E$, $\nabla_{X^1} E$, $\nabla_{Z^1} E$가

$$\nabla_{Z^2} E = F^{2\prime}(Z^2) * \nabla g$$

$$\nabla_{X^1} E = W^{2T} \nabla_{Z^2} E$$

$$\nabla_{Z^1} E = F^{1\prime}(Z^1) * \nabla_{X^1} E$$

임을 확인할 수 있다. 여기서 *는 hadamard product이다.

식 (2.5), (2.6), (2.9), (2.10)로 주어진 오류 역전파는 다음과 같이 변형된다.

$$\nabla_{W^2} E = \nabla_{Z^2} E \, X^{1T}$$

$$\nabla_{b^2} E = \mathrm{row\_sum}(\nabla_{Z^2} E)$$

$$\nabla_{W^1} E = \nabla_{Z^1} E \, X^{0T}$$

$$\nabla_{b^1} E = \mathrm{row\_sum}(\nabla_{Z^1} E)$$

여기서 $\mathrm{row_sum}$은 행렬의 row 성분의 합이다. 곧,

$$\mathrm{row\_sum} \begin{bmatrix} a_{11} \cdots a_{1n} \\ \vdots \ \ddots \ \vdots \\ a_{m1} \cdots a_{mn} \end{bmatrix} = \begin{bmatrix} a_{11} + \cdots + a_{1n} \\ \vdots \\ a_{m1} + \cdots + a_{mn} \end{bmatrix}$$

이다.

### 4.2 행 벡터 표현

프로그램을 작성하다 보면 때때로 열 벡터 대신 행 벡터를 사용하고 싶은 유혹이 생긴다. 주어진 데이터가 행 벡터 표현인 경우가 많을뿐더러 numpy의 행렬과도 잘 어울린다.

행 벡터 표현은 행렬을 전치(transpose)하여 표현하면 된다.

$$X^0 = \begin{bmatrix} x_{11}^0 \cdots x_{1p}^0 \\ \vdots \ \ddots \ \vdots \\ x_{n1}^0 \cdots x_{np}^0 \end{bmatrix}, \quad T = \begin{bmatrix} t_{11} \cdots t_{1r} \\ \vdots \ \ddots \ \vdots \\ t_{n1} \cdots t_{nr} \end{bmatrix}$$

$$Z^1 = \begin{bmatrix} z_{11}^1 \cdots z_{1q}^1 \\ \vdots \ \ddots \ \vdots \\ z_{n1}^1 \cdots z_{nq}^1 \end{bmatrix}, \ X^1 = \begin{bmatrix} x_{11}^1 \cdots x_{1q}^1 \\ \vdots \ \ddots \ \vdots \\ x_{n1}^1 \cdots x_{nq}^1 \end{bmatrix}$$

bias와 가중치 행렬도 전치시킨다.

$$b^1 = \begin{bmatrix} b_1^1 \cdots b_q^1 \end{bmatrix}, \ W^1 = \begin{bmatrix} w_{11}^1 \cdots w_{1q}^1 \\ \vdots \ \ddots \ \vdots \\ w_{p1}^1 \cdots w_{pq} \end{bmatrix}$$

행렬의 곱은 순서를 바꾼다.

$$Z^1 = b^1 + X^0 W^1, \ X^1 = F^1(Z^1), \ \ldots$$

그래디언트를 계산해 보자.

$$\nabla g = \nabla_{X^2} E = \begin{bmatrix} \frac{\partial E}{\partial x_{11}^2} \cdots \frac{\partial E}{\partial x_{1r}^2} \\ \vdots \ \ddots \ \vdots \\ \frac{\partial E}{\partial x_{n1}^2} \cdots \frac{\partial E}{\partial x_{nr}^2} \end{bmatrix}, \ F^{1\prime}(Z^1) = \begin{bmatrix} f'(z_{11}^1) \cdots f'(z_{1q}^1) \\ \vdots \ \ddots \ \vdots \\ f'(z_{n1}^1) \cdots f'(z_{nq}^1) \end{bmatrix}$$

Hadamard product는 영향을 받지 않는다.

$$\nabla_{Z^2} E = F^{2\prime}(Z^2) * \nabla g$$

행렬 곱은 순서를 바꾼다.

$$\nabla_{X^1} E = \nabla_{Z^2} E \, W^{2T}$$

이 식에서 $W^2$가 아닌 $W^{2T}$를 곱해야 한다. $W^2$가 이미 전치된 행렬이기 때문이다.

$$\nabla_{Z^1} E = F^{1\prime}(Z^1) * \nabla_{X^1} E$$

$b^k$와 $W^k$의 그래디언트는 다음과 같다.

$$\nabla_{W^2} E = X^{1T} \nabla_{Z^2} E$$

$$\nabla_{b^2} E = \text{column\_sum}(\nabla_{Z^2} E)$$

$$\nabla_{W^1} E = X^{0T} \nabla_{Z^1} E$$

$$\nabla_{b^1} E = \text{column\_sum}(\nabla_{Z^1} E)$$

column_sum은 열 성분의 합이다.

### 4.3 softmax

함수 Softmax는 categorical cross entropy와 결합하여 사용할 때가 많다. 이 때에는 (3.9)와 같이 그래디언트 계산이 단순하다. 여기서는 일반적인 경우에 대하여 알아본다.

$p$ 차원 Input 데이터 $x_1, \ldots, x_n$에 대하여

$$y_i = \text{softmax}(x_i),$$

곧, $y_{ij} = e^{x_{ij}} / \sum_{k=1}^{p} e^{x_{kj}}$라 하자. 행렬로 나타내면

$$X = \begin{bmatrix} x_1 \cdots x_n \end{bmatrix} = \begin{bmatrix} x_{11} \cdots x_{1n} \\ \vdots \ \ddots \ \vdots \\ x_{p1} \cdots x_{pn} \end{bmatrix}$$

와

$$Y = [y_1 \cdots y_n] = \begin{bmatrix} y_{11} \cdots y_{1n} \\ \vdots \ \ddots \ \vdots \\ y_{p1} \cdots y_{pn} \end{bmatrix}$$

이 된다. 식 (2.8)과 같은 일반적인 네트워크에서 $X$와 $Y$는 각각 $Z^k$와 $X^k$의 역할을 한다.

$i$ 열 $x_i$에 대하여

$$\nabla_{x_i} E = J(\mathrm{softmax})(x_i) \nabla_{y_i} E$$

를 만족한다. 식 (3.5)를 적용하여 계산하면 다음과 같다.

$$\begin{bmatrix} \frac{\partial E}{\partial x_{1i}} \\ \vdots \\ \frac{\partial E}{\partial x_{pi}} \end{bmatrix} = \begin{bmatrix} y_{1i}(1-y_{1i}) & \cdots & -y_{1i}y_{pi} \\ \vdots & \ddots & \vdots \\ -y_{pi}y_{1i} & \cdots & y_{pi}(1-y_{pi}) \end{bmatrix} \begin{bmatrix} \frac{\partial E}{\partial y_{1i}} \\ \vdots \\ \frac{\partial E}{\partial y_{pi}} \end{bmatrix}$$

$$= \begin{bmatrix} y_{1i}(1-y_{1i})\frac{\partial E}{\partial y_{1i}} - \cdots - y_{1i}y_{pi}\frac{\partial E}{\partial y_{pi}} \\ \vdots \\ -y_{pi}y_{1i}\frac{\partial E}{\partial y_{1i}} - \cdots + y_{pi}(1-y_{pi})\frac{\partial E}{\partial y_{pi}} \end{bmatrix}$$

$$= y_i * \begin{bmatrix} (1-y_{1i})\frac{\partial E}{\partial y_{1i}} - \cdots - y_{pi}\frac{\partial E}{\partial y_{pi}} \\ \vdots \\ -y_{1i}\frac{\partial E}{\partial y_{1i}} - \cdots + (1-y_{pi})\frac{\partial E}{\partial y_{pi}} \end{bmatrix}$$

$$= y_i * \left( \nabla_{y_i} E - \begin{bmatrix} y_{1i}\frac{\partial E}{\partial y_{1i}} + \cdots + y_{pi}\frac{\partial E}{\partial y_{pi}} \\ \vdots \\ y_{1i}\frac{\partial E}{\partial y_{1i}} + \cdots + y_{pi}\frac{\partial E}{\partial y_{pi}} \end{bmatrix} \right)$$

$$= y_i * \left( \nabla_{y_i} E - \left( y_{1i}\frac{\partial E}{\partial y_{1i}} + \cdots + y_{pi}\frac{\partial E}{\partial y_{pi}} \right) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \right)$$

$$= y_i * \left( \nabla_{y_i} E - \mathrm{sum}\,(y_i * \nabla_{y_i} E) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \right)$$

따라서

$$\nabla_X E = Y * \left( \nabla_Y E - \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \mathrm{column\_sum}\,(Y * \nabla_Y E) \right)$$

를 얻는다.

numpy 코드로 쓰면

```
gradXE = Y * (gradYE - np.sum(Y * gradYE, axis=0))
```

이다. 이것은 열 벡터 표현법이다. 행 벡터 표현법은

```
gradXE = Y * (gradYE - np.sum(Y * gradYE, axis=1, keepdims=True))
```

이다.