경사하강법과 역전파 (Method of gradient descent and backpropagation)

L[1]

Abstract

회귀, 특히, 선형회귀의 경우 비용함수(에러함수)의 최솟값을 찾는 대표적인 알고리즘은 경사하강법이다. 이런 경사하강법은 여러 회귀와 분류 알고리즘에서 에러 함수의 최소를 갖는 매개변수를 구하는데 사용된다. 특히 Neural Network를 훈련시키는 알고리즘인 역전파이론에서도 널리 사용된다.

1. 경사하강법

1.1. 선형회귀 : 비용함수, 특성

통계학에서 회귀란, 여러 개의 독립변수와 한 개의 종속변수 간의 상관관계를 모델링하는 기법을 통칭합니다. 다음은 Andrew Ng교수의 강의록에서 가져온 부동산 가격에 관한 데이터 입니다.

Living area (feet^2)	#bedrooms	Price (1000$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

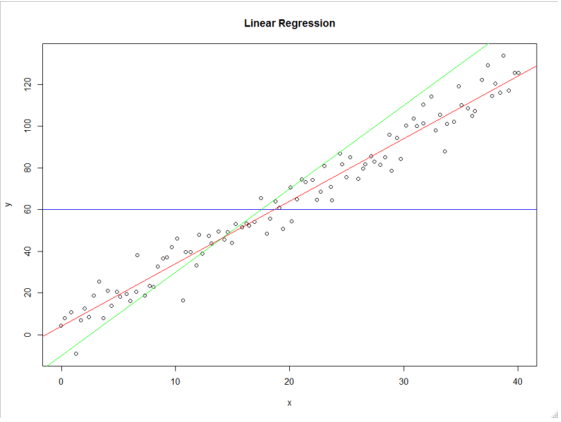
여기서 주된 관심사는 집의 넓이(x_1이라 합시다)와 침실 수(x_2)를 가지고 집의 가격(y)를 예측하는 식을 구하는 것입니다. i번째 집의 넓이, 침실 수, 가격을 각각 $x_1^{(i)}$, $x_2^{(i)}$, $y^{(i)}$라 합시다.
선형회귀란 y를 잘 근사시킬 수 있는 일차식

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2, \qquad x = (x_1, x_2)$$

을 찾는 것입니다. (즉, θ_i를 찾는 것) 여기서 θ_i들을 parameters (혹은 weights)라 하고 x_i를 특성(feature)이라 합니다. 그럼 어떻게 가장 잘 근사시키는 일차식을 찾아야 할까요? 아니, 좋은 일차 근사식은 무엇일까요?

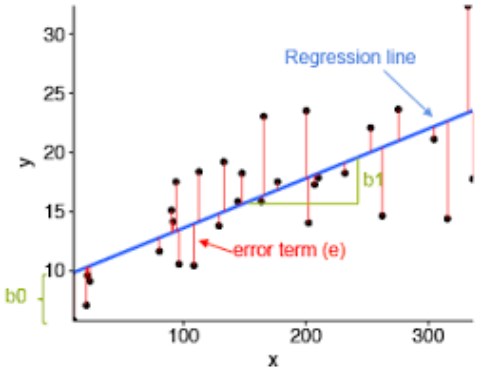
Email address: jhleepde@cau.ac.kr (L)

간단한 경우를 생각해 봅시다. 다음 주어진 데이터를 가장 잘 표현하는 식은 무엇일까요? 빨간 선, 파란 선, 녹색 선 중 어느 것일까요?



빨간 선을 고르셨죠? 빨간 선을 고른 이유는 무엇입니까?

예측값과 데이터 사이의 차이는 다음 그림의 빨간 선분의 길이로 나타납니다.



$$\sum_i |y^{(i)} - h_\theta(x^{(i)})|$$

를 최소화시키는 것은 나쁘진 않으나 절댓값 함수($|x|$)는 미분도 안되고 계산도 불편한 구석이 있습니다.

가장 유용한 답은

$$\sum_i^n (y^{(i)} - h_\theta(x^{(i)}))^2$$

을 최소화하는 겁니다. 사실 제곱을 다 더해서 루트를 씌우면 거리가 나온다는 것을 생각하면 결국

제곱합을 최소가 되도록 만드는 것은 극히 자연스럽습니다.

Definition 1. *cost(error) function is*

$$J(\theta) = \frac{1}{2}\sum_{i}^{n}(y^{(i)} - h_\theta(x^{(i)}))^2.$$

앞에 $1/2$은 생략해도 상관이 없지만(어차피 몇 배를 하더라도 최소가 되는 θ는 같습니다.) 심리적인 편안함때문에 추가한 것. (운동에너지는 $1/2mv^2$ 이고 제곱을 미분하면 서로 상쇄되기도 하고 등등의 이유로) $1/n$ 을 1/2대신 쓰기도 합니다.

위에서 $h_\theta(x) = \theta_0 + \theta_1 x_1 + \cdots + \theta_d x_d$ 입니다. $1 = x_0$라 두면 $h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_d x_d$ 로 나타납니다.
결국 우리는 이 비용함수가 최소가 되는 $\theta = (\theta_0, \theta_1, \cdots, \theta_d)$ 를 찾고자 하는 것입니다.

1.2. 경사하강법

비용함수가 최소가 되는 θ를 구하는 것은 최적화 이론(Optimization)의 중요한 주제입니다. 코쉬가 1800년대 초반에 제기한 어떤 볼록 함수의 최솟값을 찾는 방법이 바로 경사하강법입니다. 경사하강법을 쓰면 다음과 같습니다.

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta).$$

선형회귀의 경우 비용함수는 θ_j에 관한 2차식이고 아래로 볼록 함수입니다. 오른쪽의 θ_j에서 한 스텝후 왼쪽의 θ_j로 업데이트합니다. 이런 식으로 여러 번 시행한 후 $\frac{\partial}{\partial\theta_j}J(\theta)$가 0에 가까워지면 더이상 업데이트를 멈추고 스톱합니다.

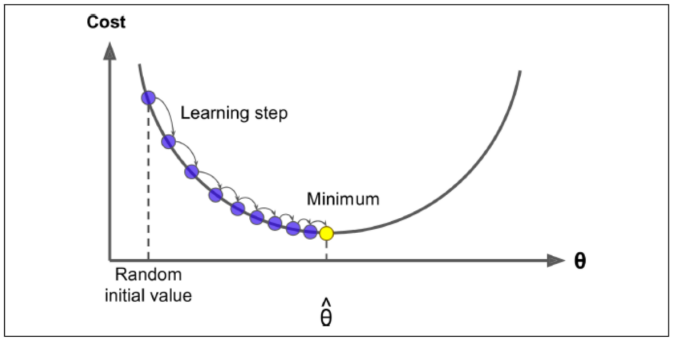


Figure 4-3. Gradient Descent

위의 그림을 따라 머리속으로 경사하강법을 생각해봅시다. 만약 처음에 게스한 θ가 $\hat{\theta}$보다 작다면

$\frac{\partial}{\partial\theta}J(\theta)$ 는 접선의 기울기를 생각하면 음수입니다. α는 학습률로 양수인 한 스텝의 크기입니다. 따라서 그 다음 스텝에 약간 오른쪽으로 이동합니다. 이와 같이 하다 보면 최소점에 가까워지고 가까워지면 $\frac{\partial}{\partial\theta}J(\theta)$ 값이 0에 가까워집니다. 처음에 게스한 θ가 $\hat{\theta}$보다 크면 $\frac{\partial}{\partial\theta}J(\theta)$가 양수이고 스텝마다 점점 왼쪽으로 갑니다.

이제 계산을 해봅시다. 편의상 training set이 하나 (x, y)라 합시다.

$$\begin{aligned}\frac{\partial}{\partial\theta_j}J(\theta) &= \frac{\partial}{\partial\theta_j}\frac{1}{2}(h_\theta(x)-y)^2 \\ &= 2\cdot\frac{1}{2}(h_\theta(x)-y)\cdot\frac{\partial}{\partial\theta_j}(h_\theta(x)-y) \\ &= (h_\theta(x)-y)\cdot\frac{\partial}{\partial\theta_j}\left(\sum_{i=0}^{d}\theta_i x_i - y\right) \\ &= (h_\theta(x)-y)x_j.\end{aligned}$$

경사하강법에 대입하면 training set이 하나인 경우

$$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}.$$

이를 LMS update rule (LMS: least mean square) 혹은 Widrow-Hoff learning rule이라 한다.

training set이 여러 개인 경우 (실제 문제에서는 수 만개 이상일 수 있습니다.) 방법은 크게 다음 두 가지로 나누어진다.

1) batch gradient descent(BGD) : 업데이트 한번 할 때마다 전체 데이터 셋에 대한 에러를 구한 뒤 미분하여 모델의 parameter를 업데이트 하는 방법.(전체 업데이트 횟수는 적고 한번 업데이트에 비용이 많이 든다. 만약 비용함수가 최소가 아닌 극소가 있을 때, 즉 볼록함수가 아니면 극소값에서 빠져 나오지 못한다.)

Repeat until convergence : {

$$\theta_j := \theta_j + \alpha\sum_{i=1}^{n}(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}, \qquad \text{for every } j,$$

}

2) stochastic gradient descent(SGD) : 추출된 데이터 한 개에 대해서 error gradient 를 계산하여 업데이트 하는 방법 .(업데이트 횟수는 많아지나 업데이트 한번 비용은 싸다. 극소에서 벗어나기

쉬우나 최소점으로 수렴하지는 않고 그 근처로만 간다.)

Loop : {

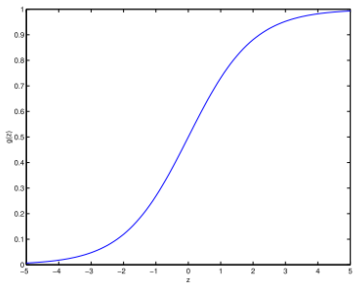
$$\theta_j := \theta_j + \alpha(y^{(i)} - h_\theta(x^{(i)}))x_j^{(i)}, \qquad \text{for every } j,$$

}

1.3. 분류

분류는 회귀와 함께 지도학습의 두가지 큰 줄기 중 하나입니다. 주어진 데이터로부터 0(실패)과 1(성공) 두 가지로 분류하는 것을 생각하겠습니다. 무지 많은 분류 알고리즘이 있지만 여기서는 비용함수와 경사하강법을 소개하는 것이니 로지스틱 회귀만을 소개하려고 합니다. 선형 회귀를 분류 알고리즘에 사용하려면 어떤 문제가 있을까요? 분류, 이진 분류는 y값을 0또는 1로 분류하는 겁니다. 그런데 선형 회귀에서는 0과 1은 커녕 $y = \theta^T \cdot x$는 항상 양의 무한대와 음의 무한대로 가까이 가는 값들이 존재합니다. 그래서 사람들이 생각한 것이 다음 sigmoid 함수 (또는 logistic 함수라고 함) 입니다.

$$g(z) = \frac{1}{1 + e^{-z}}.$$



시그모이드 함수의 역함수는

$$f(z) = \ln\left(\frac{z}{1-z}\right)$$

인 log-odds 함수 입니다. 또 시그모이드 함수의 도함수는 다음 식을 만족합니다.

$$g'(z) = \frac{1}{(1+e^{-z})^2}(e^{-z}) = \frac{1}{(1+e^{-z})}\left(1 - \frac{1}{1+e^{-z}}\right)$$

$$= g(z)(1 - g(z)).$$

Logistic regression에서는 다음과 같이 근사 함수를 변화시킵니다.

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}.$$

그럼 어떻게 θ를 고를까요?
경사하강법을 사용합시다. 그럼 비용함수를 정의해야 합니다. 확률이 다음과 같다고 가정하는 것은 그럴 듯 해 보입니다. (진짜 그럴 듯 해보이나요?)

$$P(y = 1|x, \theta) = h_\theta(x), \quad P(y = 0|x, \theta) = 1 - h_\theta(x)$$

$h_\theta(x)$라는 것은 결국 1일 확률로 해석하는 것입니다. 그럼 이제 비용함수를 정의하겠습니다. 앞의 식으로부터 다음 확률 밀도 함수는 그럴 듯 해보입니다.(진짜 그럴 듯 해보이길 바랍니다. 아래를 베르누이 분포의 확률밀도함수라고 합니다.)

$$p(y|x, \theta) = (h_\theta(x))^y (1 - h_\theta(x))^{1-y}.$$

만약 n개의 sample 이 있다면, likelihood(일어날 가능성) 함수는

$$L(\theta) = \prod_{i=1}^{n} p(y^{(i)}|x^{(i)}, \theta) = \prod_{i=1}^{n} (h_\theta(x^{(i)}))^{y^{(i)}} (1 - h_\theta(x^{(i)}))^{1-y^{(i)}}$$

인데, 이 likelihood 함수의 최대가 된다는 것은 예측이 틀리는 상황을 최소화하는 것입니다. 일반적으로 log를 취한 log-likelihood 함수의 최댓값을 구하는 것이 쉽습니다. 비용함수는 $J(\theta) := -\log L(\theta)$로 정의합니다. 일반적으로 이러한 분류 문제의 비용함수를 entropy 형태의 함수라고 합니다.

$$J(\theta) = -\sum_{i=1}^{n} (y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))).$$

위 식의 해석은 $h_\theta(x^{(i)})$ 는 i 번째 샘플이 1일 추정확률이고 $y^{(i)}$는 실제 0또는 1의 레이블입니다. 가장 극단적으로 틀린 경우를 생각해 봅시다. 만약 $y^{(i)} = 1$ 인 어떤 경우를 $h_\theta(x^{(i)}) = 0$으로 추측했다면 $y^{(i)} \log h_\theta(x^{(i)}) = -\infty$ 가 됩니다. 비용함수앞에 마이너스 부호가 있으니 전체 비용

함수는 양의 무한대가 되겠죠. 거꾸로 가장 극단적으로 예측을 정확하게 맞췄다면, 즉, $y^{(i)} = 1$ 인 경우 $h_\theta(x^{(i)}) = 1$로, $y^{(i)} = 0$ 인 경우 $h_\theta(x^{(i)}) = 0$의 결과를 얻었다면 우리의 비용함수는 0이 됩니다. 그래서 우리는 비용함수가 최소가 되는 매개변수를 찾기를 원합니다. (그렇다고 모델을 너무 세분화하여 무작정 비용을 0에 가깝게 만드는 모델을 생각하면 과대적합이 일어나서 실제 예측이 좋지 않게 됩니다.)

경사하강법을 사용해 보겠습니다.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

계산의 편의상 이번에도 sample이 하나 (x, y)라고 하겠습니다.

$$J(\theta) = -\log L(\theta) = -y \log h_\theta(x) - (1-y) \log(1 - h_\theta(x))$$

$$= -y \log g(\theta^T x) - (1-y) \log(1 - g(\theta^T x)).$$

$$\frac{\partial}{\partial \theta_j} J(\theta) = \left(-y \frac{1}{g(\theta^T x)} + (1-y) \frac{1}{1 - g(\theta^T x)} \right) \frac{\partial}{\partial \theta_j} g(\theta^T x)$$

$$= \left(-y \frac{1}{g(\theta^T x)} + (1-y) \frac{1}{1 - g(\theta^T x)} \right) g(\theta^T x)(1 - g(\theta^T x)) \frac{\partial}{\partial \theta_j} (\theta^T x)$$

$$= (-y(1 - g(\theta^T x)) + (1-y) g(\theta^T x)) x_j = -(y - h_\theta(x)) x_j.$$

경사하강법에 대입하면

$$\theta_j := \theta_j + \alpha (y - h_\theta(x)) x_j$$

를 얻습니다.

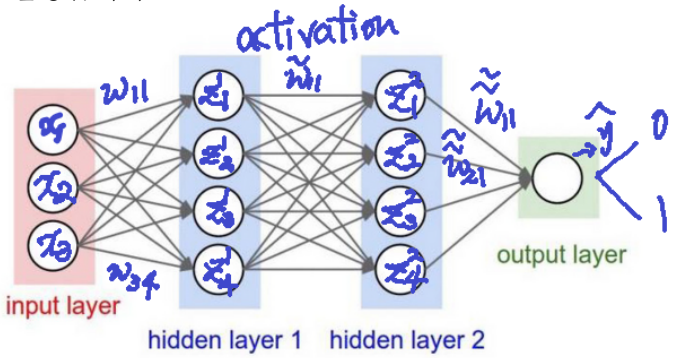
2. Backpropagation

2.1. 신경망 개요

신경망(Neural Network)은 머신러닝 알고리즘 중 하나이고 많이 쓰이는 알고리즘 중 하나입니다. (그렇다고 이 알고리즘이 어떤 데이터에 대하여 더 좋은 정확도를 가지고 있다고 말할 수는 없습니다. 알려져 있는 이론은 일반적인 데이터에 모두 좋은 결과를 내는 알고리즘이란 없다는

것입니다.)

일단 신경망의 동작 원리에 대하여 간략하게 설명하고자 합니다. 편의상 분류를 하는 신경망에 대하여 설명합시다.



1) input layer (입력층) : i번째 데이터 $(x^{(i)}, y^{(i)})$ 에 대하여 입력층 $x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, \cdots, x_d^{(i)}$ (d는 특성수) 을 생각합니다.(그림엔 안그렸지만 $x_0 = 1$도 생각합니다.) 편의상 (i)는 생략하겠습니다. 이를 hidden layer(은닉층) 1에 전달하는데 각각 x_i에 중요도(weight)를 곱해서 전달합니다. 예를 들어 다음과 같습니다.

$$z_k^1 = \sum_{j=1}^{d} w_{jk} x_j + w_{0k}, \quad k = 1, \cdots, m$$

m 은 은닉층 1의 노드 숫자라고 하겠습니다.

2) hidden layer (은닉층) 1 : 입력받은 z_k^1를 활성화 함수에 대입하여 변환합니다. 활성화함수는 여러 가지 종류가 있지만 대표적인 것은 다음 세 가지입니다. 시그모이드 함수, tangent hyperbolic 함수, ReLU 함수 입니다. 여기서는 편의상 시그모이드 함수를 생각하겠습니다.

$$a_k^1 = g(z_k^1), \qquad k = 1, ..., m$$

이를 다시 각 a_j^1에 weight를 곱하여 다음 은닉층에 전달합니다. (물론 $a_0^1 = 1$로 생각하여 절편을 표현합니다.) 예를 들어 다음과 같습니다.

$$z_k^2 = \sum_{j=1}^{m} w_{jk}^1 a_j^1 + w_{0k}^1, \quad k = 1, \cdots, n$$

n 은 은닉층 2의 노드 숫자라고 하겠습니다.

3) hidden layer (은닉층) 2 : 은닉층 1과 똑같습니다. 입력받은 z_k^2를 활성화 함수에 대입하여

변환합니다.

$$a_k^2 = g(z_k^2), \qquad k = 1, ..., n$$

다시 각 a_j^2에 weight를 곱하여 다음 은닉층에 전달합니다. (물론 $a_0^2 = 1$로 생각하여 절편을 표현합니다.) 여기서는 은닉층의 갯수가 2개고 바로 다음 output layer(출력층)에 전달한다고 하겠습니다. 그리고 출력층의 노드가 하나인 경우만 생각하지요. 예를 들어 다음과 같습니다.

$$s = \sum_{j=1}^{n} w_j^2 a_j^2 + w_0^2.$$

4) output layer (출력층) : 입력받은 s를 activation 함수에 대입하여 $\hat{y} = g(s)$를 얻습니다. 만약 g가 시그모이드 함수의 경우라면 어떤 데이터가 1일 확률이 $\hat{y}$로 예측된 것이고 0.5이상이면 1로 판정합니다.

5) 비용함수 또는 에러 함수 : 엔트로피 형태의 에러 함수

$$J(\theta) = -\sum_{i=1}^{n} (y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})))$$

를 생각하면 되는데 다음과 같습니다. (이번에도 편의상 데이터의 갯수는 하나 (x, y)로...)

$$J = -y \log \hat{y} - (1 - y) \log(1 - \hat{y}) = -y \log g(s) - (1 - y) \log(1 - g(s)).$$

우리는 훈련데이터를 통하여 이 비용함수가 최소가 되는 매개변수를 찾을 겁니다. 여기서 매개변수는? 여러 가지 있겠지만 주된 매개변수는 요인의 weight들입니다. 그런데 문제는 비용함수는 한 가지인데 weight는 처음 입력층에서 이야기한 w_{jk}, 은닉층 1에서 2로 넘어갈 때 w_{jk}^1, 은닉층 2에서 출력층으로 넘어갈 때의 w_j^2 등 다양할 뿐 아니라 위의 비용함수의 식에 explicit하게 표현되지도 않았습니다. 그러므로 여기에 경사하강법을 사용하려면 숨어있는 weight들에 의한 변화율을 계산해야 하므로 연쇄법칙(chain rule)이 필요합니다.

2.2. Backpropagation

일변수 함수의 연쇄 법칙은 다음과 같습니다. $f(x) = u$, $g(u) = y$ 일 때,

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = g'(f(x))f'(x)$$

Proposition 1. *(Chain Rule) Suppose that $z = f(x, y)$ is differnetiable function of x and y, where $x = g(s, t)$ and $y = h(s, t)$ are both differentiable functions of s and t. Then z is differentiable function of s and t and*

$$\frac{\partial z}{\partial s} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial s},\ \frac{\partial z}{\partial t} = \frac{\partial f}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial t}$$

Example

If $z = e^x \sin y$, where $x = st^2$ and $y = s^2t$, find $\frac{\partial z}{\partial s}$ and $\frac{\partial z}{\partial t}$.

Proposition 2. *Chain Rule(General Case) Suppose that $z = f(x_1, \cdots, x_n)$ is differnetiable function of $x_1, \cdots, x_n$, where $x_j = g_j(t_1, \cdots t_m)$ $(j = 1, \cdots, n)$ are differentiable functions of $t_1, \cdots, t_m$. Then z is differentiable function of $t_1, \cdots, t_m$ and*

$$\frac{\partial z}{\partial t_i} = \frac{\partial f}{\partial x_1}\frac{\partial x_1}{\partial t_i} + \cdots + \frac{\partial f}{\partial x_n}\frac{\partial x_n}{\partial t_i}$$

이제 앞의 문제에 대하여 경사하강법을 해보도록 하겠습니다. 일단 참고로 다시 비용 함수와 weight를 하나 써두지요.

$$J(s) = -y \log g(s) - (1 - y) \log(1 - g(s)), \quad g(s) = \frac{1}{1 + e^{-s}}, \quad s = \sum_{j=0}^{n} w_j^2 a_j^2$$

경사하강법은

$$w_j^2 := w_j^2 - \alpha \frac{\partial}{\partial w_j^2} J(s).$$

Step 1. $\frac{\partial}{\partial w_j^2} J(s)$ 계산하기.

$$\frac{\partial}{\partial w_j^2} J(s) = \frac{\partial J}{\partial s}\frac{\partial s}{\partial w_j^2}$$

여기서

$$\frac{\partial J}{\partial s} = -\frac{y}{g(s)} g'(s) - (1 - y)\frac{-g'(s)}{1 - g(s)}$$

$$= -\frac{y}{g(s)} g(s)(1 - g(s)) + \frac{(1 - y)}{1 - g(s)} g(s)(1 - g(s))$$

$$= -y(1 - g(s)) + (1 - y)g(s) = -y + g(s).$$

한편, $\frac{\partial s}{\partial w_j^2} = a_j^2$ 이므로,

$$\frac{\partial}{\partial w_j^2} J(s) = -(y - g(s))a_j^2.$$

출력층에 연결된 w_j^2는 업데이트하는 계산이 어렵지 않네요!

Step 2. $\frac{\partial}{\partial w_{jk}^1} J(s)$ 구하기.

이제 머리가 아프기 시작합니다. $s = \sum_{j=0}^{n} w_j^2 a_j^2$ 이고, $a_j^2 = g(z_j^2)$, 그리고

$$z_j^2 = \sum_{i=0}^{m} w_{ij}^1 a_i^1, \quad j = 1, \cdots, n$$

입니다.

$$\frac{\partial}{\partial w_{jk}^1} J(s) = \frac{\partial J}{\partial s} \frac{\partial s}{\partial w_{jk}^1}$$

$\frac{\partial J}{\partial s}$는 이미 계산했으므로,

$$\frac{\partial s}{\partial w_{jk}^1} = \sum_{l=0}^{m} \frac{\partial s}{\partial a_l^2} \frac{\partial a_l^2}{\partial z_l^2} \frac{\partial z_l^2}{\partial w_{jk}^1}$$

를 계산하면 됩니다.(복잡하지요?) 각각은

$$\frac{\partial s}{\partial a_l^2} = w_l^2,$$

$$\frac{\partial a_l^2}{\partial z_l^2} = g(z_l^2)(1 - g(z_l^2)) = a_l^2(1 - a_l^2),$$

$$\frac{\partial z_l^2}{\partial w_{jk}^1} = \begin{cases} a_j^1, & l = k, \\ 0 & l \neq k \end{cases}$$

이를 종합하면

$$\frac{\partial}{\partial w_{jk}^1} J(s) = -(y - g(s))w_k^2 a_k^2 (1 - a_k^2) a_j^1$$

가 됩니다.

Step 3. $\frac{\partial}{\partial w_{jk}} J(s)$ 구하기.

머리가 아프긴 하지만 이 계산을 못할 것은 없습니다. 앞에서 $a_i^1 = g(z_i^1)$이고 $z_i^1 = \sum_{j=0}^{d} w_{ji} x_j$

이므로 다시 한번 연쇄법칙을 쓰면 됩니다. 하지만 과정 자체가 지난 스텝의 반복이므로 여기서는 생략하도록 하겠습니다.

이와 같은 과정을 통해 경사하강법으로 매개변수를 업데이트 해나갑니다. 지금 본 것처럼 모델에 대해서 거꾸로 돌아가는 연쇄법칙 과정이 있기 때문에 이를 backpropagation 이라고 합니다.