2022.03.10

Week 2
Optimization
N.N Architecture

김민수, 박상용

CH4.
Finding Minima Algorithm

KIM MINSU(ESC, Yonsei University)

4.2 Gradient Descent

Gradient descent를 이해하기 위해 2가지 개념을 이해해야함.

1) Level Sets

2) Directional derivative(방향 도함수)

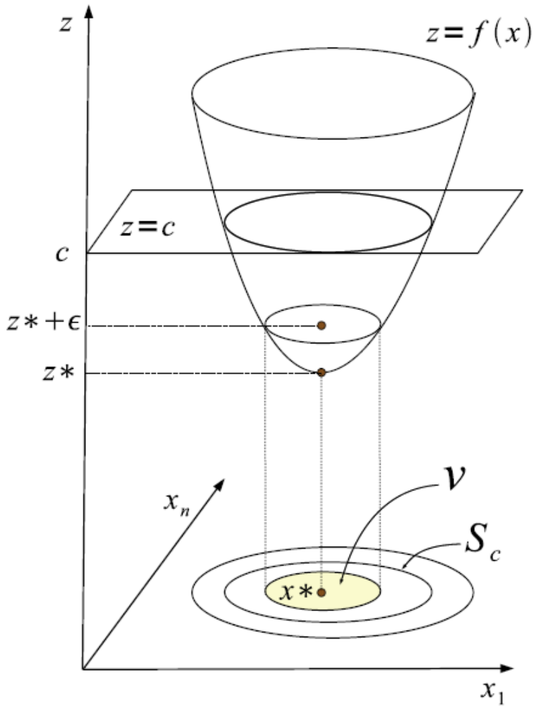
1. Level sets

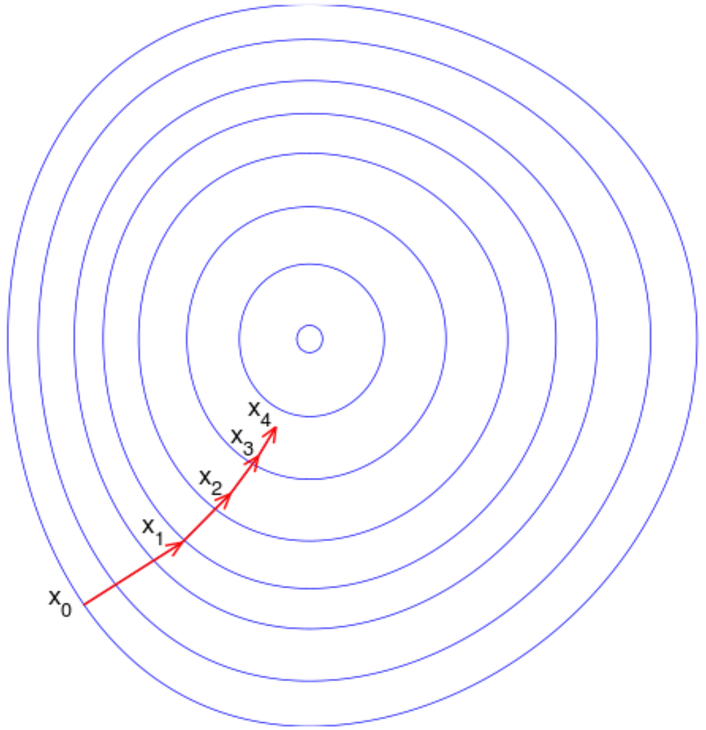
4.2.1 Level sets

Consider the function $z = f(x)$, $x \in D \subset \mathbb{R}^n$, with $n \geq 2$ and define the set $\mathcal{S}_c = f^{-1}(\{c\}) = \{x \in \mathbb{R}^n; f(x) = c\}$. Assume the function f is differentiable with nonzero gradient, $\nabla f(x) \neq 0$, $x \in \mathcal{S}_c$. Under this condition, $\mathcal{S}_c$ becomes an $(n-1)$-dimensional hypersurface in $\mathbb{R}^n$. The family $\{\mathcal{S}_c\}_c$ is called the *level hypersurfaces* of the function f. For $n = 2$ they are known under the name of *level curves*. Geometrically, the level hypersurfaces are obtained intersecting the graph of $z = f(x)$ with horizontal planes $\{z = c\}$, see Fig. 4.1.

Level sets : 다변수 함수의 함수값이 같은 집합

1. Level sets





2. Directional derivative(방향 도함수)

Another concept used later is the *directional derivative*, which measures the instantaneously the rate of change of a function at a point in a given direction. More precisely, let v be a unitary vector in $\mathbb{R}^n$ and consider the differentiable function $f : \mathcal{U} \subset \mathbb{R}^n \to \mathbb{R}$. The directional derivative of f at the point $x^0 \in \mathcal{U}$ is defined by

$$\frac{\partial f}{\partial v}(x^0) = \lim_{t \searrow 0} \frac{f(x^0 + tv) - f(x^0)}{t}.$$

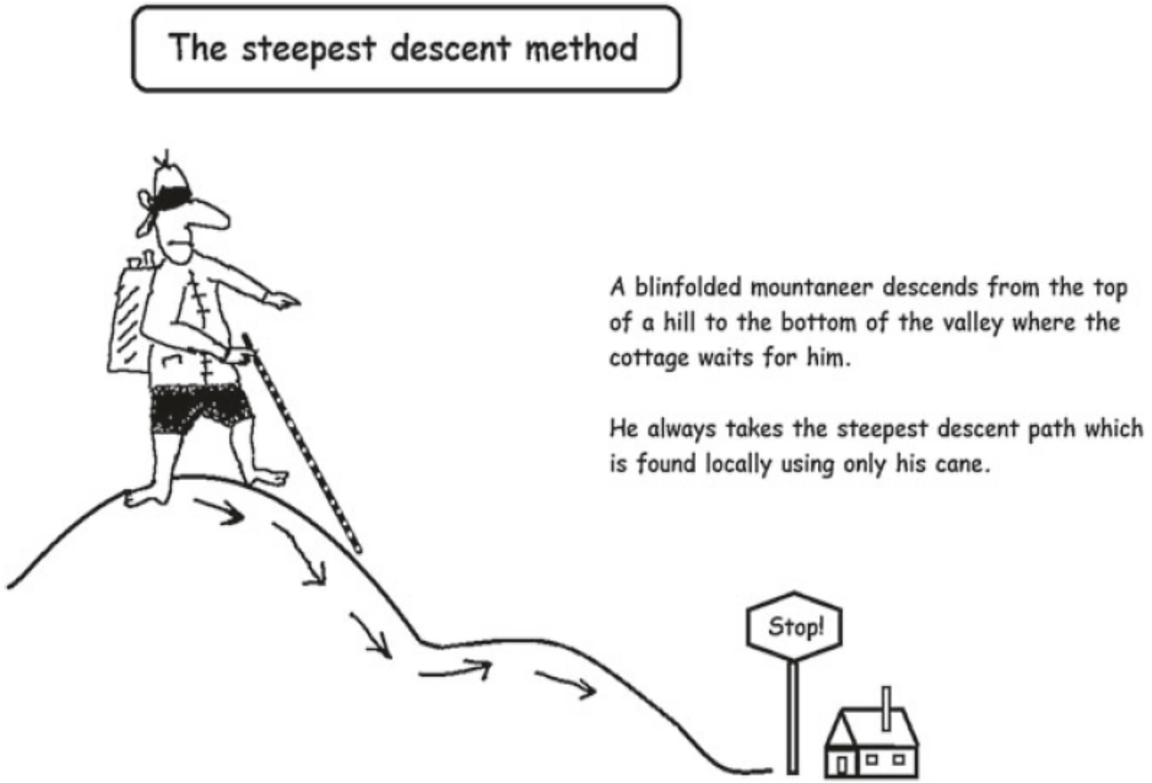
2. Directional derivative(방향 도함수)

Note that partial derivatives with respect to coordinates, $\frac{\partial f}{\partial x_k}$, are directional derivatives with respect to the coordinate vectors $v = (0, \ldots, 1, \ldots, 0)^T$. An application of chain rule provides a computation of the directional derivative as a scalar product:

$$\begin{aligned} \frac{\partial f}{\partial v}(x^0) = \frac{d}{dt} f(x^0 + tv)\Big|_{t=0+} &= \sum_{k=1}^{n} \frac{\partial f}{\partial x_k}(x^0 + tv)\, v^k \Big|_{t=0} \\ &= v^T \nabla f(x^0) = \langle \nabla f(x^0), v \rangle. \end{aligned}$$

partial derivatives는 directional derivatives의 special case이다.
방향도함수를 **내적**의 형태로 표현가능!

본격적으로 Gradient Descent에 대해 살펴보겠습니다!



Gradient Descent

In order to apply this method, we are interested in finding the unitary direction v, in which the function f decreases as much as possible within a given small step size η. The change of the function f between the value at the initial point x^0 and the value after a step of size η in the direction v is written using the linear approximation as

$$\begin{aligned} f(x^0 + \eta v) - f(x^0) &= \sum_{k=1}^{n} \frac{\partial f}{\partial x_k}(x^0)\eta v^k + o(\eta^2) \\ &= \eta\langle \nabla f(x^0), v\rangle + o(\eta^2). \end{aligned}$$

Gradient Descent

In the following, since η is small, we shall neglect the effect of the quadratic term $o(\eta^2)$. Hence, in order to obtain v such that the change in the function has the largest negative value, we shall use the Cauchy inequality for the scalar product

$$-\|\nabla f(x^0)\|\,\|v\| \le \langle \nabla f(x^0), v\rangle \le \|\nabla f(x^0)\|\,\|v\|.$$

It is known that the inequality on the left is reached for vectors that are negative proportional.[3] Since $\|v\| = 1$, the minimum occurs for

$$v = -\frac{\nabla f(x^0)}{\|\nabla f(x^0)\|}.$$

Then the largest change in the function is approximately equal to

$$f(x^0 + \eta v) - f(x^0) = \eta\langle \nabla f(x^0), v\rangle = -\eta\|\nabla f(x^0)\|.$$

Gradient Descent

The algorithm consists of the following iteration that constructs the following sequence (x^n):

(i) Choose an initial point x^0 in the basin of attraction of the global minimum x^*.

(ii) Construct the sequence $(x^n)_n$ using the iteration

$$x^{n+1} = x^n - \eta \frac{\nabla f(x^n)}{\|\nabla f(x^n)\|}. \tag{4.2.3}$$

This guarantees a negative change of the objective function, which is given by $f(x^{n+1}) - f(x^n) = -\eta \|\nabla f(x^n)\| < 0$.

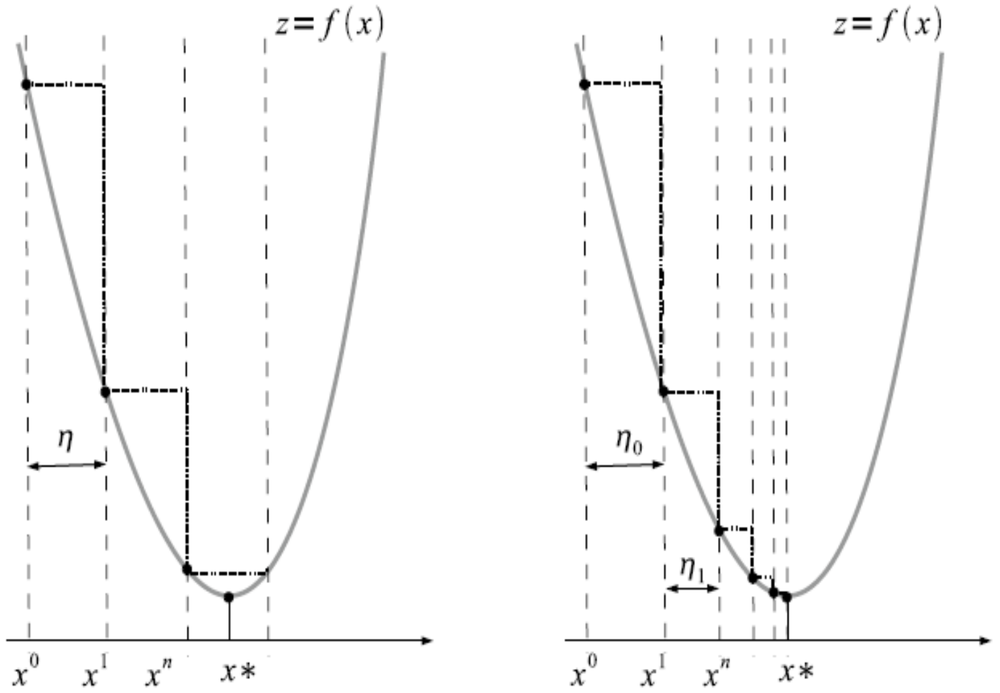


Figure 4.7: **a**. *The use of a fixed learning rate η leads to missing the minimum x^*.* **b**. *An adjustable learning rate η_n provides a much better approximation of the minimum. In this case the descent amount is proportional with the slope of the curve at the respective points; these slopes become smaller as we approach the critical point x^*.*

변형된 형태의 Gradient Descent

기존 Gradient Descent는 fixed learning rate를 사용하여 문제가 있음.

이러한 문제를 해결하기 위해 learning rate를 **2가지 형태**로 변형함.

1) learning rate를 gradient 크기와 비례하게 설정. $\eta_n = \delta \|\nabla f(x^n)\|$

4.2.3식에 대입하면 $x^{n+1} = x^n - \delta \nabla f(x^n)$. 도출된다.

과연 이러한 형태의 algorithm은 수렴할 것인가? p.84-85에 증명이 나와있습니다!

변형된 형태의 Gradient Descent(2)

2) Line Search Method

This is a variant of the method of steepest descent with an adjustable learning rate η. The rate is chosen as in the following. Starting from an initial point x^0, consider the normal direction on the level hypersurface $\mathcal{S}_{f(x^0)}$ given by the gradient $\nabla f(x^0)$. We need to choose a point, x^1, on the line given by this direction at which the objective function f reaches a minimum. This is equivalent with choosing the value $\eta_0 > 0$ such that

$$\eta_0 = \arg\min_{\eta} f(x^0 - \eta \nabla f(x^0)). \tag{4.2.5}$$

변형된 형태의 Gradient Descent(2)

2) Line Search Method

The procedure continues with the next starting point $x^1 = x^0 - \eta_0 \nabla f(x^0)$. By an iteration we obtain the sequence of points (x^n) and the sequence of learning rates (η_n) defined recursively by

$$\begin{aligned} \eta_n &= \arg\min f(x^n - \eta \nabla f(x^n)) \\ x^{n+1} &= x^n - \eta_n \nabla f(x^n). \end{aligned}$$

The method of line search just described has the following geometric significance. Consider the function

$$g(\eta) = f(x^0 - \eta \nabla f(x^0)),$$

and differentiate it to get

$$g'(\eta) = -\langle \nabla f(x^0 + \eta \nabla f(x^0), \nabla f(x^0) \rangle. \qquad (4.2.6)$$

변형된 형태의 Gradient Descent(2)

2) Line Search Method

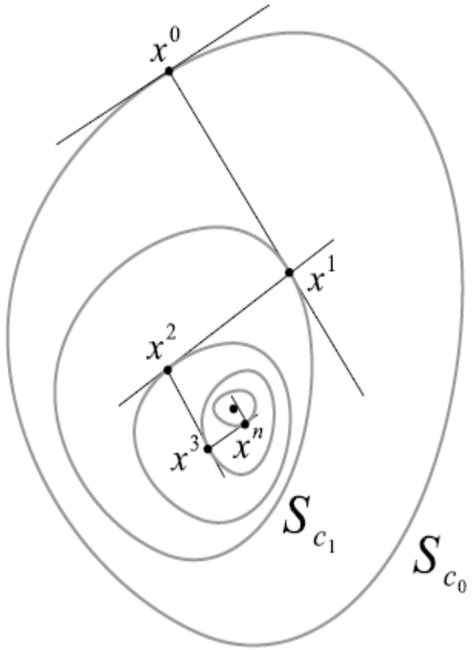


Figure 4.10: *The approximation of the minimum x^* by the method of line search.*

4.4 Momentum Method

왜 moment를 도입했을까?

In order to avoid getting stuck in a local minimum of the cost function, several methods have been designed. The basic idea is that shaking the system by adding extra velocity or energy will make the system to pass over the energy barrier and move into a state of lower energy.[5]

The *momentum method* modifies the gradient descent by introducing a velocity variable and having the gradient modify the velocity rather than the position. It is the change in velocity that will affect the position. Besides the learning rate, this technique uses an extra hyperparameter, which models the friction, which reduces gradually the velocity and has the ball rolling toward a stable equilibrium, which is the bottom of the cup. The role of this method is to accelerate the gradient descent method while performing the minimization of the cost function.

local minimum에 stuck하는 문제를 해결하기 위함!

Classical Momentum Method

The classical momentum method (Polyak, 1964, see [98]) provides the following simultaneous updates for the position and velocity

$$x^{n+1} = x^n + v^{n+1} \tag{4.4.16}$$

$$v^{n+1} = \mu v^n - \eta \nabla f(x^n), \tag{4.4.17}$$

It is worth noting that for $\mu \to 0$, the previous model recovers the familiar model of the gradient descent, $x^{n+1} = x^n - \eta \nabla f(x^n)$.

어떤 조건에서 converge?

Convergence Condition(4.4.2)

Iterating equation (4.4.16)

$$\begin{aligned} x^n &= x^{n-1} + v^n \\ x^{n-1} &= x^{n-2} + v^{n-1} \\ \cdots &= \cdots \\ x^1 &= x^0 + v^1 \end{aligned}$$

and then adding, yields the expression of the position in terms of velocities

$$x^n = x^0 + \sum_{k=1}^{n} v^k. \tag{4.4.28}$$

Denote for simplicity $b_n = \nabla f(x^n)$. Iterating equation (4.4.17), we obtain

$$\begin{aligned} v^n &= \mu v^{n-1} - \eta b_{n-1} \\ &= \mu(\mu v^{n-2} - \eta b_{n-2}) - \eta b_{n-1} \\ &= \mu^2 v^{n-2} - \mu\eta b_{n-2} - \eta b_{n-1} \\ &= \mu^2(\mu v^{n-3} - \eta b_{n-3}) - \mu\eta b_{n-2} - \eta b_{n-1} \\ &= \mu^3 v^{n-3} - \mu^2\eta b_{n-3} - \mu\eta b_{n-2} - \eta b_{n-1}. \end{aligned}$$

$$v^{n+1} = \mu^{n+1} v^0 - \eta \sum_{i=0}^{n} \mu^{n-i} b_i. \tag{4.4.29}$$

Proposition 4.4.4 *Let $(a_n)_n$ be a sequence of real numbers convergent to 0 and $\sum_{n\geq 0} b_n$ be an absolute convergent numerical series. Then*

$$\lim_{n\to\infty} \sum_{i=0}^{n} a_i b_{n-i} = 0.$$

Theorem 4.4.5 *Consider two numerical series $\sum_{n\geq 0} a_n$ and $\sum_{n\geq 0} b_n$, one convergent and the other absolute convergent. Then their convolution series is convergent and its sum is equal to the product of the sums of the given series, i.e.*

$$\sum_{n\geq 0} \Big(\sum_{i=0}^{n} a_i b_{n-i} \Big) = \Big(\sum_{n\geq 0} a_n \Big) \Big(\sum_{n\geq 0} b_n \Big).$$

Convergence Condition(4.4.2)

Proposition 4.4.6 *(a) If $\nabla f(x^n)$ converges to 0, then the sequence $(v^n)_n$ is also convergent to 0, as $n \to \infty$.*
(b) Assume the series $\sum_{n\geq 0} \|\nabla f(x^n)\|$ is convergent. Then both sequences $(x^n)_n$ and $(v^n)_n$ are convergent.

Proof: (a) Recall that $0 < \mu < 1$. Since (b_n) converges to 0 and the geometric series $\sum_{n\geq 0} \mu^n$ is absolute convergent, Proposition 4.4.4 implies $\lim_{n\to\infty} \sum_{i=0}^{n} \mu^{n-i} b_i = 0$. Since μ^n is convergent to 0, taking the limit in (4.4.29) we obtain

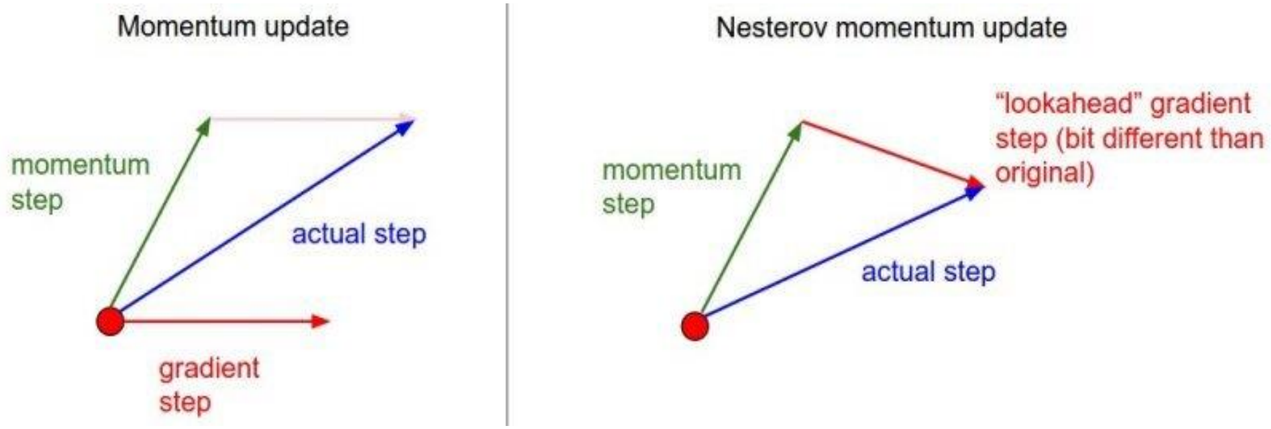
$$v^* = \lim_{n\to\infty} v^{n+1} = v^0 \lim_{n\to\infty} \mu^{n+1} - \eta \lim_{n\to\infty} \sum_{i=0}^{n} \mu^{n-i} b_i = 0.$$

$$\begin{aligned} x^{n+1} &= x^0 + \sum_{k=1}^{n+1} v^k = x^0 + \sum_{k=0}^{n} v^{k+1} \\ &= x^0 + \sum_{k=0}^{n} \Big[\mu^{k+1} v^0 - \eta \sum_{i=0}^{k} \mu^{k-i} b_i\Big] \\ &= x^0 + v^0 \frac{\mu(1-\mu^{n+1})}{1-\mu} - \eta \sum_{k=0}^{n} \sum_{i=0}^{k} \mu^{k-i} b_i. \end{aligned}$$

$$\begin{aligned} x^* &= \lim_{n\to\infty} x^{n+1} = x^0 + v^0 \frac{\mu}{1-\mu} - \eta \Big(\sum_{n\geq 0} \mu^n\Big)\Big(\sum_{n\geq 0} b_n\Big) \\ &= x^0 + v^0 \frac{\mu}{1-\mu} - \frac{\eta}{1-\mu} \sum_{n\geq 0} \|\nabla f(x^n)\|. \end{aligned}$$

Nesterov Accelerated Gradient

변형된 형태의 momentum method



Remark 4.4.7 An improvement of the classical momentum method has been proposed by Nesterov [91]. This is obtained by modifying the momentum method the argument of the gradient; instead of computing it at the current position x^n, it is evaluated at the corrected value $x^n + \mu v^n$:

$$x^{n+1} = x^n + v^{n+1} \tag{4.4.30}$$

$$v^{n+1} = \mu v^n - \eta \nabla f(x^n + \mu v^n). \tag{4.4.31}$$

4.5 Adagrad

Adagrad

- Adagrad가 등장한 이유

“ 안 가본 곳은 빠르게, 많이 가본 곳은 천천히 세밀하게 탐색” 의 컨셉

fixed learning rate를 사용하기 보다는 각 **스텝**마다, **변수**마다 다른 lr를 사용하고자 함.

$$G_t = \sum_{\tau=1}^{t} g_\tau g_\tau^T$$

$$x(t+1) = x(t) - \eta G_t^{-1/2} g_t,$$

하지만 이 방식은 Gt^(-1/2)를 구하는 부분이 high dimension에서는 상당히 비효율적임.
대신 대각원소의 합을 이용하여 update를 진행한다.

Adagrad

$$v(t) = v(t-1) + g_{t-1}^2$$
$$x(t+1) = x(t) - \eta \frac{g_t}{\sqrt{|v(t)|}}$$

- Adagrad의 문제점

v(t)가 이전 gradient의 제곱값이 계속하여 더해지기 때문에 증가한다.

v(t)가 커짐에 따라 분모에 크기가 커지고 update의 크기는 점점 작아진다.
=> 효과적인 학습이 이루어지지 않게 된다. (RMSProp의 등장)

4.6 RMSProp

RMSProp

- Adagrad를 보완한 부분

기울기를 균일하게 더하는 것이 아니라, 과거의 기울기는 비중을 줄이고 새로운 기울기의 비중을 늘리는 지수이동평균(Exponential Moving Average)를 사용.

If $C(x)$ denotes the cost function, and $g_t = \nabla C(x(t))$ is its gradient evaluated at time step t, then the running average is defined recursively by

$$v(t) = \gamma v(t-1) + (1-\gamma) g_{t-1}^2,$$

$$x(t+1) = x(t) - \eta \frac{g_t}{\sqrt{|v(t)|}},$$

4.7 Adam

Adam

- **Momentum**과 **RMSProp**의 방식을 **융합**한 알고리즘.

For this, we denote the gradient of the cost function at the time step t by $g_t = \nabla C(x(t))$. We consider two exponential decay rates for the moment estimates, $\beta_1, \beta_2 \in [0, 1)$, fix an initial vector $x(0) = x_0$, initialize the moments by $m(0) = 0$ and $v(0) = 0$, and consider the moments updates

$$\begin{aligned} m(t) &= \beta_1 m(t-1) + (1-\beta_1) g_t \\ v(t) &= \beta_2 v(t-1) + (1-\beta_2)(g_t)^2, \end{aligned}$$

m(t) : biased estimates of the first moment(mean) of the gradient
v(t) : biased estimates of the second moment(uncentered variance) of the gradient

Adam에서는 m,v의 initial value가 0이기 때문에 학습 초반에 m,v가 0으로 biased되는 문제가 발생. => correction이 필요.

Adam

- **Momentum**과 **RMSProp**의 방식을 **융합**한 알고리즘.

$$
\begin{aligned}
m(t) &= (1-\beta_1)\sum_{i=1}^{t}\beta_1^{t-i}g_i \\
v(t) &= (1-\beta_2)\sum_{i=1}^{t}\beta_2^{t-i}(g_i)^2.
\end{aligned}
$$

$$
\begin{aligned}
\mathbb{E}[m(t)] &= (1-\beta_1)\sum_{i=1}^{t}\beta_1^{t-i}\mathbb{E}[g_i] = (1-\beta_1^t)\mathbb{E}[g_t] \\
\mathbb{E}[v(t)] &= (1-\beta_2)\sum_{i=1}^{t}\beta_2^{t-i}\mathbb{E}[[(g_i)^2] = (1-\beta_2^t)\mathbb{E}[(g_t)^2].
\end{aligned}
$$

Therefore, the bias-corrected moments are

$$
\begin{aligned}
\widehat{m}(t) &= m(t)/(1-\beta_1^t) \\
\widehat{v}(t) &= v(t)/(1-\beta_2^t).
\end{aligned}
$$

The final recursive formula is given by

$$
x(t+1) = x(t) - \eta\frac{\widehat{m}(t)}{\sqrt{|\widehat{v}(t)|}+\epsilon},
$$

with $\epsilon > 0$ a small scalar used to prevent division by zero. Some default settings for the aforementioned hyperparameters used in [31] are $\eta = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.99$, and $\epsilon = 10^{-8}$.

4.10 Increasing Resolution Method

Increasing Resolution Method

local minima가 많을 때 유용한 algorithm, differentiable 할 필요도 X

The idea is to blur the signal $y = f(x)$ using a Gaussian filter, $G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$, obtaining the signal $f_\sigma(x)$, which is given by the convolution

$$f_\sigma(x) = (f * G_\sigma)(x) = \int_{\mathbb{R}} f(u) G_\sigma(x-u)\, du = \int_{\mathbb{R}} f(x-u) G_\sigma(u)\, du.$$

The positive parameter σ controls the signal resolution. A large value of σ provides a signal $f_\sigma(x)$, which retains a rough shape of $f(x)$. On the other side, a small σ means to retain details. Actually using the fact that the Gaussian tends to the Dirac measure, $\lim_{\sigma \searrow 0} G_\sigma(x) = \delta(x)$, we have

$$\lim_{\sigma \searrow 0} f_\sigma(x) = \lim_{\sigma \searrow 0} (f * G_\sigma)(x) = (f * \delta)(x) = f(x).$$

This means that for small values of σ the filtered signal is very close to the initial signal, fact which agrees with the common sense.

Increasing Resolution Method

Gaussian Filter을 이용하여 signal을 rough하게 만들고 rough한 signal에서 minimum 부근을 찾는다. 이 과정을 몇 번 반복하여 global minimum을 찾는다.

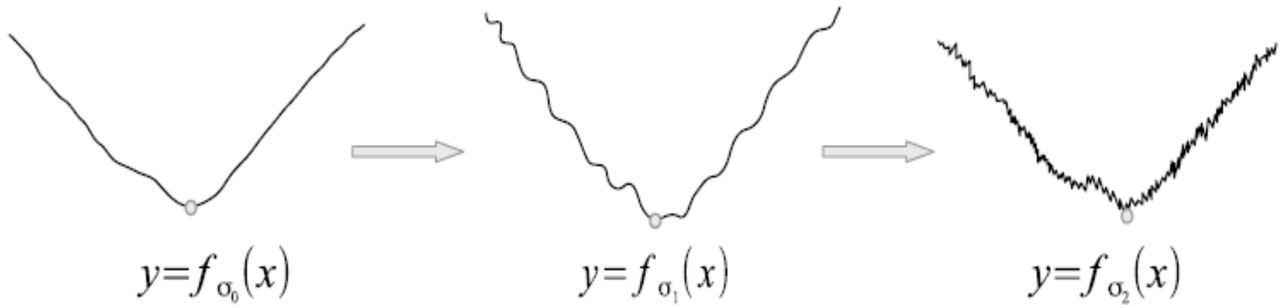


Figure 4.17: *The increase in the signal resolution. The minimum on each resolution profile is represented by a little circle.*

Increasing Resolution Method

Consider the resolution schedule $\sigma_0 > \sigma_1 > \cdots > \sigma_k > 0$. Then $f_{\sigma_0}, \ldots, f_{\sigma_k}$ are different resolutions levels of the signal $y = f(x)$, see Fig. 4.17. We shall start with a large value of σ_0. Then the signal $y = f_{\sigma_0}(x)$ looks like a rough shape of $y = f(x)$, and it does not have any local minima if σ_0 is large enough. Let

$$x_0 = \arg\min_x f_{\sigma_0}(x).$$

Since f_{σ_0} is smooth and x_0 is the only minimum, it can be achieved using the gradient descent method.

We consider the next resolution signal $y = f_{\sigma_1}(x)$ and apply the gradient descent method, starting from x_0, to obtain the lowest value in a neighborhood $\mathcal{V}_0$ of x_0 at

$$x_1 = \arg\min_{x \in \mathcal{V}_0} f_{\sigma_1}(x).$$

Next, we apply the gradient descent on a neighborhood $\mathcal{V}_1$ of x_1 for the signal f_{σ_2} and obtain

$$x_2 = \arg\min_{x \in \mathcal{V}_1} f_{\sigma_2}(x).$$

We continue the procedure until we reach

$$x_k = \arg\min_{x \in \mathcal{V}_{k-1}} f_{\sigma_k}(x).$$

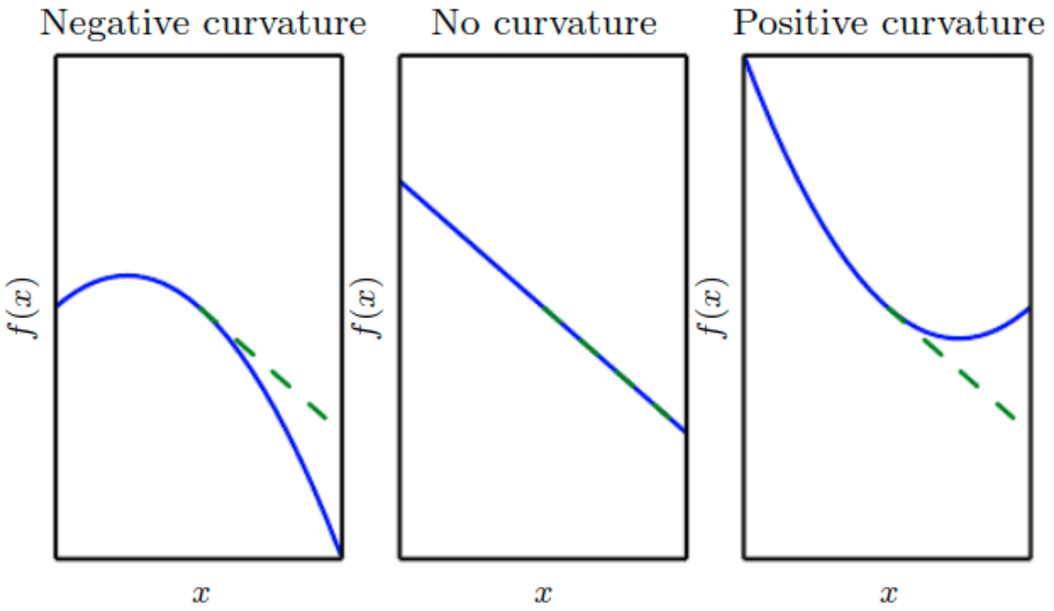
Due to the increase in resolution we have the descending sequence of neighborhoods $\mathcal{V}_0 \supset \cdots \supset \mathcal{V}_{k-1}$. Each one contains an approximation of the global minimum. If the schedule is correctly chosen, then x_k is a good approximation of the global minimum of $f(x)$.

4.11 Hessian Method

Hessian Method

기존의 방법들은 **곡률(curvature)**을 고려하지 않았다.

$$\mathbf{H}_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \, \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \, \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \, \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \, \partial x_1} & \frac{\partial^2 f}{\partial x_n \, \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix},$$



Hessian Method

The gradient methods presented in the previous sections do not take into account the curvature of the surface $z = f(x)$, which is described by the matrix of second partial derivatives

$$H_f(x) = \left(\frac{\partial^2 f(x)}{\partial x_i \partial x_j} \right)_{i,j},$$

called *Hessian*. The eigenvalues of the matrix H_f will be denoted by $\{\lambda_i\}$, $1 \leq i \leq n$. The corresponding eigenvectors, $\{\xi_j\}_j$, satisfy $H_f \xi_i = \lambda_i \xi_i$, with $\|\xi_i\| = 1$.

Proposition 4.11.1 *Let f be a real-valued function, twice continuous differentiable. The following hold:*

(a) The Hessian matrix is symmetric;

(b) For any x, the eigenvalues of $H_f(x)$ are real;

(c) Assume H has distinct eigenvalues. Then for any nonzero vector v, we have

$$\frac{\langle H_f(x)v, v\rangle}{\|v\|^2} = \sum_{i=1}^{n} w_j \lambda_j,$$

where $\{\lambda_j\}$ are the eigenvalues of $H_f(x)$ and w_j are weights, with $w_j \geq 0$ and $\sum_{j=1}^{n} w_j = 1$.

(d) Let $\lambda_{\min}$ and $\lambda_{\max}$ be the largest and the smallest eigenvalue of $H_f(x)$, respectively. Then

$$\lambda_{\min}\|v\|^2 \leq \langle H_f(x)v, v\rangle \leq \lambda_{\max}\|v\|^2.$$

Hessian : negative definite

Formula (4.2.4) provides the recurrence in the case of gradient descent method

$$x^{n+1} = x^n - \delta \nabla f(x^n). \tag{4.11.33}$$

$$f(x^{n+1}) \approx f(x^n) + (x^{n+1} - x^n)^T \nabla f(x^n) + \frac{1}{2}(x^{n+1} - x^n)^T H_f(x^n)(x^{n+1} - x^n)$$

$$= f(x^n) - \delta \|\nabla f(x^n)\|^2 + \frac{1}{2}\delta^2 \langle H_f(x^n)\nabla f(x^n), \nabla f(x^n)\rangle.$$

The last term on the right is the correction term due to the curvature of f. There are two distinguished cases to look into:

1. The Hessian H_f is negative definite (all eigenvalues are negative). In this case the correction term is negative

$$\frac{1}{2}\delta^2 \langle H_f(x^n)\nabla f(x^n), \nabla f(x^n)\rangle < 0.$$

The previous Taylor approximation implies the inequality

$$f(x^{n+1}) < f(x^n) - \delta \|\nabla f(x^n)\|^2 < f(x^n),$$

i.e., for any learning rate $\delta > 0$ each iteration provides a lower value for f than the previous one.

Hessian : positive definite

2. The Hessian H_f is positive definite (all eigenvalues are positive). In this case the correction term is positive

$$\frac{1}{2}\delta^2\langle H_f(x^n)\nabla f(x^n), \nabla f(x^n)\rangle > 0.$$

Proposition 4.11.1, part (d), provides the following margins of error for the correction term

$$0 < \frac{1}{2}\delta^2\lambda_{\min}\|\nabla f(x^n)\|^2 \leq \frac{1}{2}\delta^2\langle H_f(x^n)\nabla f(x^n), \nabla f(x^n)\rangle \leq \frac{1}{2}\delta^2\lambda_{\max}\|\nabla f(x^n)\|^2.$$

The idea is that if the correction term is too large, then the inequality $f(x^{n+1}) < f(x^n)$ might not hold. Considering the worst-case scenario, when the term is maximum, the following inequality has to be satisfied:

$$f(x^n) - \delta\|\nabla f(x^n)\|^2 + \frac{1}{2}\delta^2\lambda_{\max}\|\nabla f(x^n)\|^2 < f(x^n).$$

This is equivalent with the following condition on the learning rate

$$\delta < \frac{2}{\lambda_{\max}}. \tag{4.11.34}$$

Newton's Method

$$f(x) \approx F(x) = f(x^0) + (x - x^0)^T \nabla f(x^0) + \frac{1}{2}(x - x^0)^T H_f(x^0)(x - x^0),$$

where $H_f(x^0)$ is the Hessian matrix of f evaluated at x^0. Since f is convex, its Hessian is positive definite. Therefore, the quadratic function F has a minimum at the critical point x^*, which satisfies $\nabla F(x) = 0$. Using that $\nabla F(x) = \nabla f(x^0) + H_f(x^0)(x - x^0)$, the equation $\nabla F(x) = 0$ becomes

$$H_f(x^0)x = -\nabla f(x^0) + H_f(x^0)x^0.$$

Assuming the Hessian does not have any zero eigenvalues, then it is invertible and we obtain the following critical point:

$$x^* = x^0 - H_f^{-1}(x^0)\nabla f(x^0),$$

Newton's Method

Newton's method consists of using this formula iteratively constructing the sequence $(x^n)_{n\geq 1}$ defined recursively by

$$x^{n+1} = x^n - H_f^{-1}(x^n)\nabla f(x^n). \qquad (4.12.35)$$

This method converges to the minimum of f faster than the gradient descent method. However, despite its powerful value, it has a major weakness, which is the need to compute the inverse of the Hessian of f at each approximation.

Newton's Method

Remark 4.12.1 It is worth noting that in the case $n = 1$ the previous iterative formula becomes

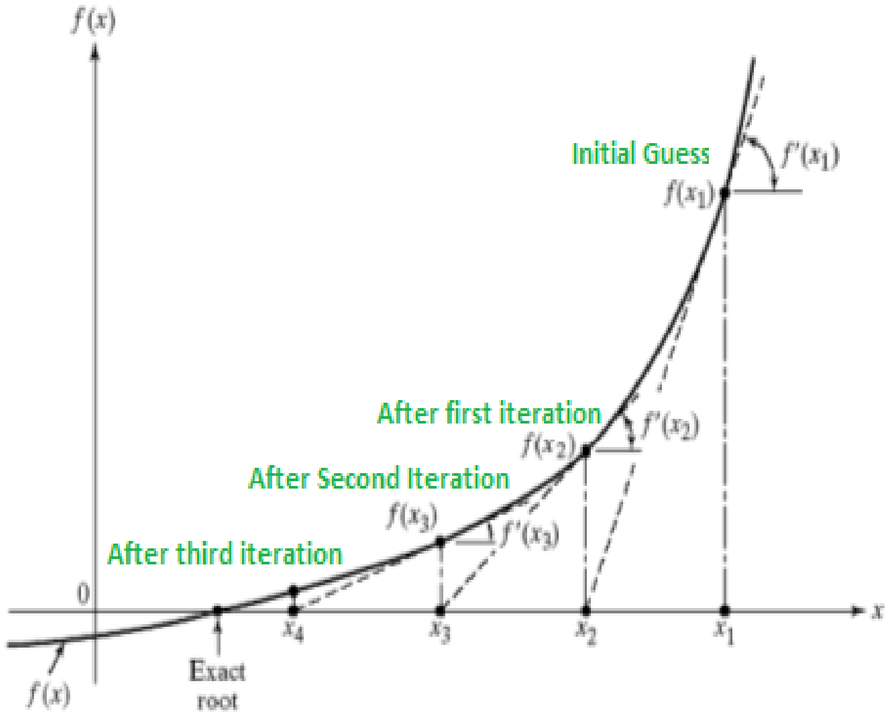
$$x^{n+1} = x^n - \frac{f'(x^n)}{f''(x^n)}.$$

If we denote $h(x) = f'(x)$, then looking for the critical point of f is equivalent to searching for the zero of $h(x)$. Convexity of f implies that h is increasing, which guarantees the uniqueness of the zero. The previous iteration formula is written as

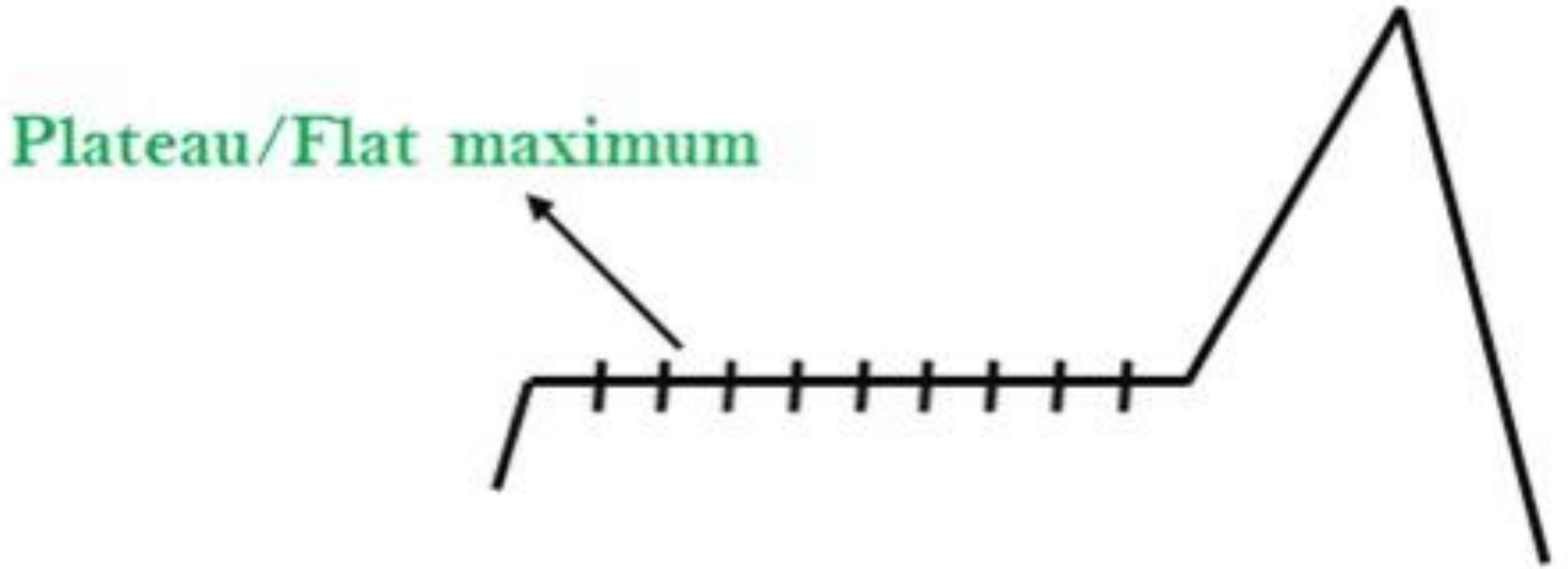
$$x^{n+1} = x^n - \frac{h(x^n)}{h'(x^n)}.$$

We have arrived now to the familiar Newton-Raphson method for searching a zero of h using successive tangent lines.

Newton's Method



Stochastic Search



Objective : 저 구간을 빠져나가자!

Stochastic Search

Ito diffusion process

$$\begin{aligned} dX_t &= b(X_t)dt + \sigma(X_t)dW_t \\ X_0 &= x_0, \end{aligned}$$

drift dispersion

$$\varphi(t) = \mathbb{E}[f(X_t)|X_0 = x_0]$$

Stochastic Search

Definition D.8.1 *A Brownian motion is a stochastic process* W_t, $t \geq 0$, *which satisfies:*
(i) $W_0 = 0$ *(the process starts at the origin);*
(ii) *if* $0 \leq u < t < s$, *then* $W_s - W_t$ *and* $W_t - W_u$ *are independent (the process has independent increments);*
(iii) $t \to W_t$ *is continuous;*
(iv) *the increments are normally distributed, with* $W_t - W_s \sim \mathcal{N}(0, |t-s|)$.

Note also that $\mathbb{E}[W_t] = 0$, $\mathbb{E}[W_t^2] = t$ and $Cov(W_t, W_s) = \min\{s, t\}$.

Stochastic Search

Dynkin's formula

$$\mathbb{E}^x[f(X_t)] = f(x) + \mathbb{E}^x\left[\int_0^t \mathcal{A}f(X_s)\,ds\right]$$

$$\mathbb{E}^x[f(X_t)] = \mathbb{E}[f(X_t)|X_0 = x]$$

$$\mathcal{A}f(x) = \lim_{t \searrow 0} \frac{\mathbb{E}^x[f(X_t)] - f(x)}{t}$$

Stochastic Search

$$\mathcal{A} = \frac{1}{2}\sum_{i,j}(\sigma\sigma^T)_{ij}\frac{\partial^2}{\partial x_i \partial x_j} + \sum_k b_k \frac{\partial}{\partial x_k} = \frac{1}{2}\sigma\sigma^T\nabla^2 + \langle \nabla, b \rangle$$

$$\begin{aligned} \varphi(t) &= f(x_0) + \int_0^t \mathbb{E}[\mathcal{A}\big(f(X_s)\big)|X_0 = x_0]\, ds \\ \varphi(t+dt) &= f(x_0) + \int_0^{t+dt} \mathbb{E}[\mathcal{A}\big(f(X_s)\big)|X_0 = x_0]\, ds \end{aligned}$$

Stochastic Search

$$\begin{aligned}\Delta\varphi(t) &= \varphi(t+dt) - \varphi(t) = \int_t^{t+dt} \mathbb{E}[\mathcal{A}\big(f(X_s)\big)|X_0 = x_0]\, ds \\ &= \mathbb{E}[\mathcal{A}\big(f(X_t)\big)|X_0 = x_0] + o(dt^2).\end{aligned}$$

Objective

$$\Delta\varphi < 0 \quad \rightarrow \quad \mathbb{E}[\mathcal{A}\big(f(X_t)\big)|X_0 = x_0] < 0$$

Stochastic Search

Objective

$$Find\ \sigma(x)\ \ \&\ \ b(x)$$

$$In\ a\ plateau, Hessian\ is\ small\ !$$

$$\Rightarrow When\ Hessian\ is\ small, the\ diffusion\ is\ large!!$$

$$(\sigma\sigma^T)_{ij} = \lambda(H_f)^{-1}_{ij}$$

Stochastic Search

(i) f is of class C^2;

(ii) f is strictly convex

$\Rightarrow$ *Hessian is real, symmetric, positive definite, and nondegenerate*

$\Rightarrow$ *Hessian is invertible* $\quad \therefore \exists\, \sigma\ s.t. \sigma\sigma^T = \lambda H_f^{-1}$

$$\begin{aligned} \mathcal{A}f(x) &= \frac{1}{2}\lambda \sum_{i,j} (H_f^{-1})_{ij} (H_f)_{ij} + \sum_k b_k(x) \frac{\partial f}{\partial x_k} \\ &= \frac{n}{2}\lambda + \langle \nabla f(x), b(x) \rangle \end{aligned}$$

Stochastic Search

$$\begin{aligned}\mathcal{A}f(x) &= \frac{1}{2}\lambda\sum_{i,j}(H_f^{-1})_{ij}(H_f)_{ij} + \sum_k b_k(x)\frac{\partial f}{\partial x_k} \\ &= \frac{n}{2}\lambda + \langle\nabla f(x), b(x)\rangle\end{aligned}$$

$$Minimize < \nabla f, b > \ \rightarrow \ \mathrm{b(x)} = -\eta\nabla f(x)$$

$$\mathcal{A}f(x) = \frac{n}{2}\lambda - \eta\|\nabla f(x)\|^2$$

Stochastic Search

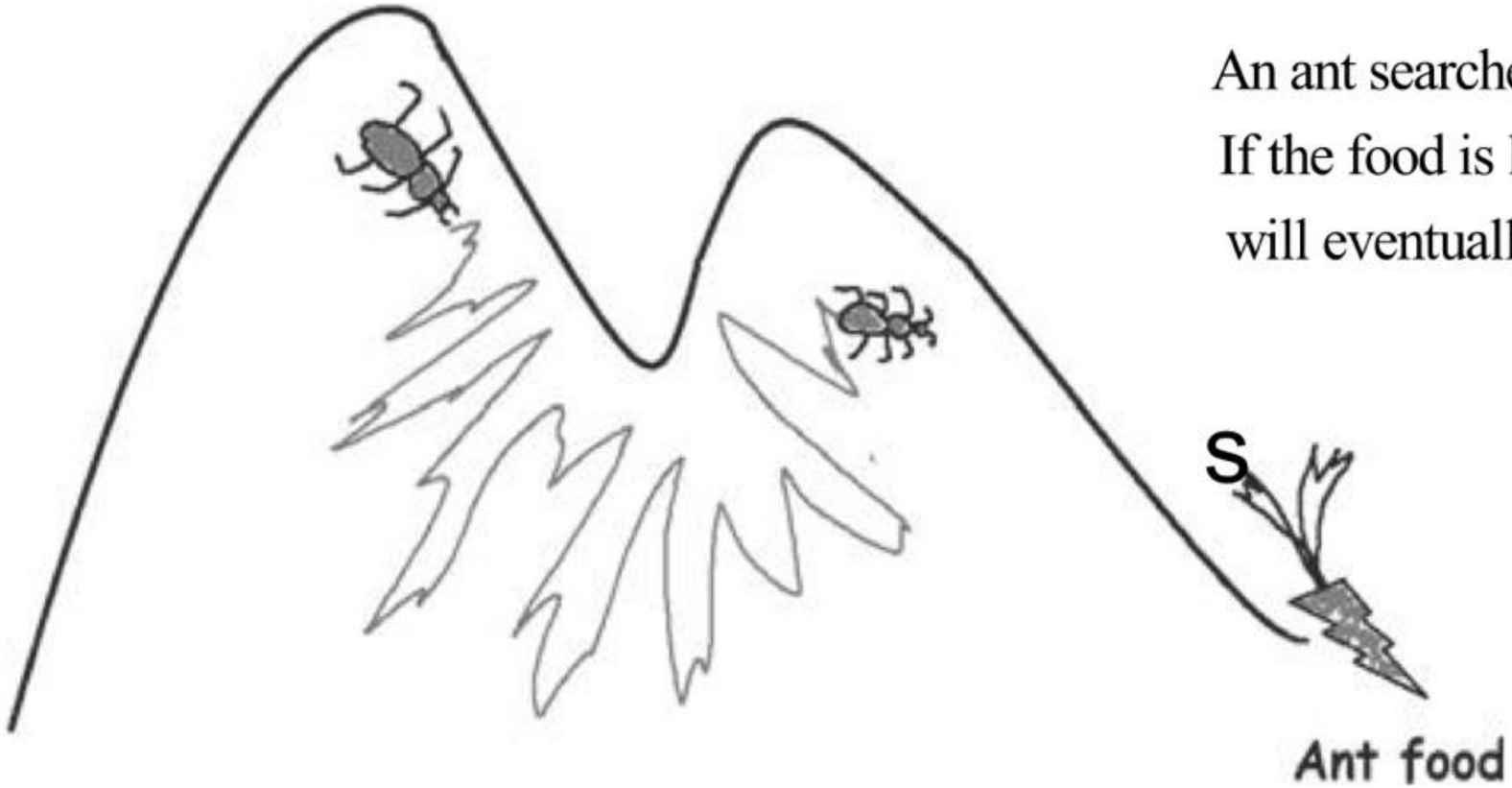
$$\mathcal{A}f(x) < 0$$

$$\frac{\lambda}{\eta} < \frac{2}{n}\|\nabla f(x)\|^2$$

$$L = \inf_x \left|\|\nabla f(x)\|\right|^2 \qquad \lambda = \frac{2L}{n}\eta$$

$$\mathbb{E}[\mathcal{A}\big(f(X_t)\big)|X_0 = x_0] < 0$$

Stochastic Search

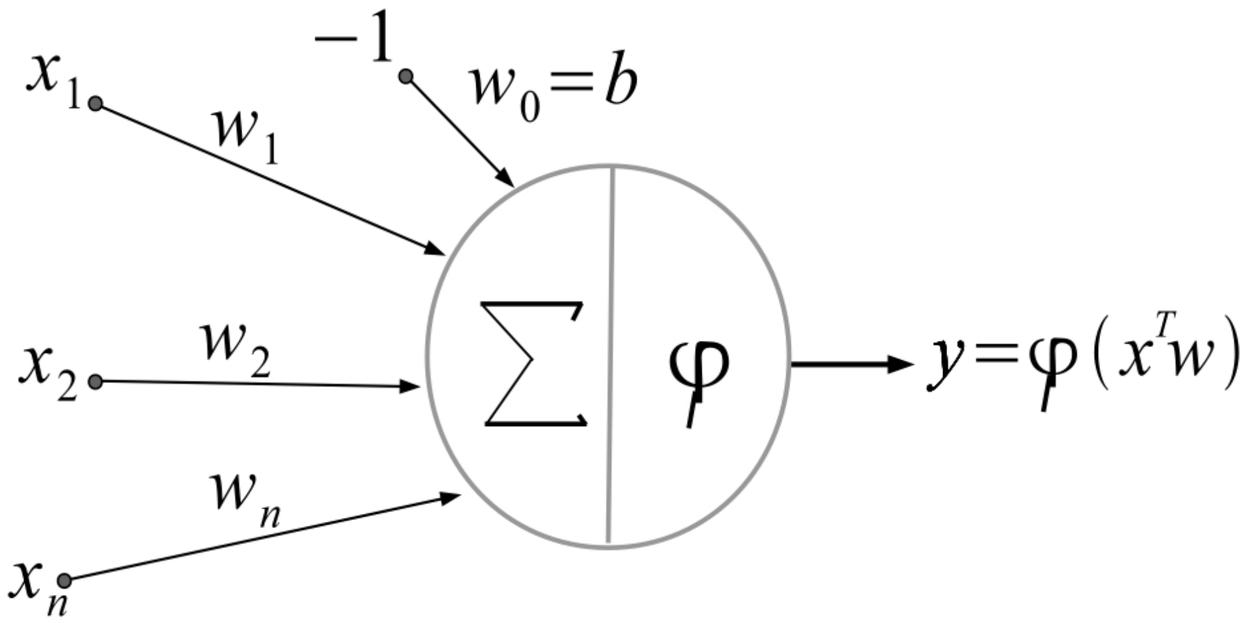


An ant searches for food in a stochastic way. If the food is located in the valley, the ant will eventually find it.

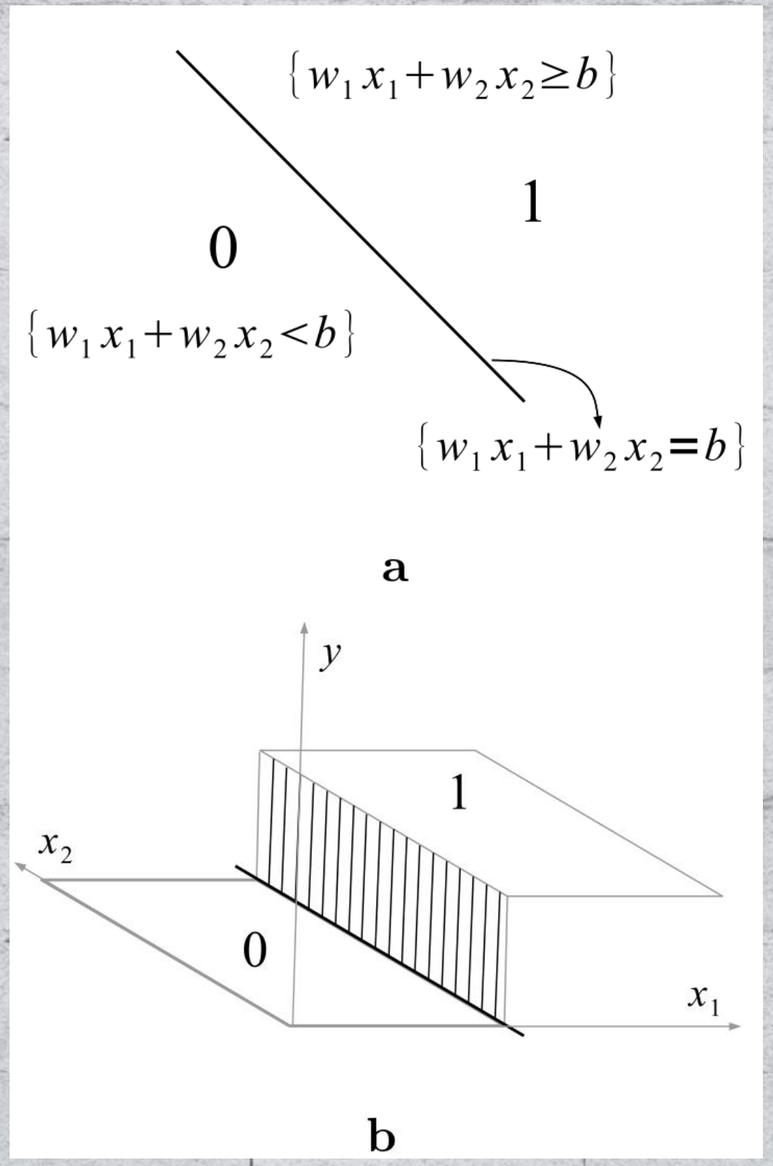
CH5. Neurons

Definition 5.1.1

An abstract neuron is a quadruple $(\mathrm{x}, \mathrm{w}, \varphi, y)$, *where* $x^T = (x_0, x_1, \ldots, x_n)$ *is the input vector,* $w^T = (w_0, w_1, \ldots, w_n)$ *is the weights vector, with* $x_0 = -1$ *and* $w_0 = b$*, the bias, and* φ *is an activation function that defines the outcome function* $y = \varphi(x^T w) = \varphi(\sum_i^n w_i x_i)$.



$$A\ perceptron\ is\ a\ neuron\ with\ the\ input\ x_i \in \{0,1\}$$
$$,and\ with\ the\ activation\ function\ given\ by\ the\ Heaviside\ function$$
$$\varphi(x) = \begin{cases} 0 & if\ x < 0 \\ 1 & if\ x \geq 0 \end{cases}$$
$$The\ output\ of\ the\ perceptron\ is\ given\ as\ a\ threshold\ gate$$
$$y = \varphi(x^T w - b) = \begin{cases} 0\ \ if\ \sum_i^n w_i x_i < b \\ 1\ \ if\ \sum_i^n w_i x_i \geq b \end{cases}$$



Geometric Interpretation Of a Perceptron

$$H = \{(x_1, \dots, x_n); \sum_i^n w_i x_i = b\}$$

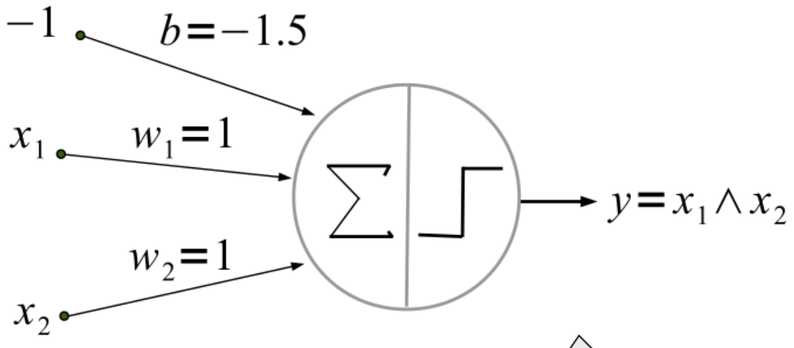
$$N^T = (w_1, \dots, w_n)$$

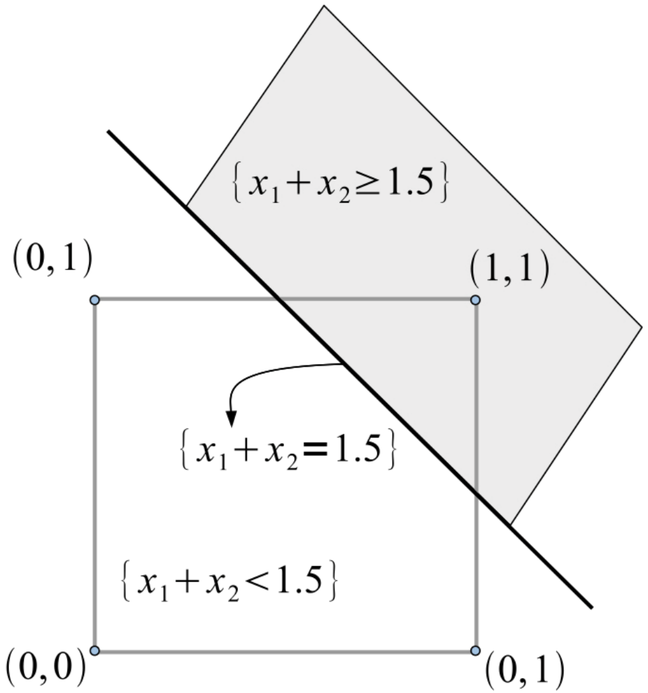
x_1	x_2	$y = x_1 \wedge x_2$
0	0	0
0	1	0
1	0	0
1	1	1

x_1	x_2	$y = x_1 \vee x_2$
0	0	0
0	1	1
1	0	1
1	1	1

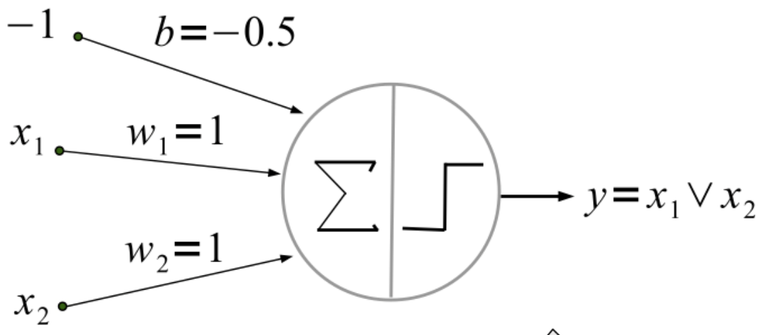
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

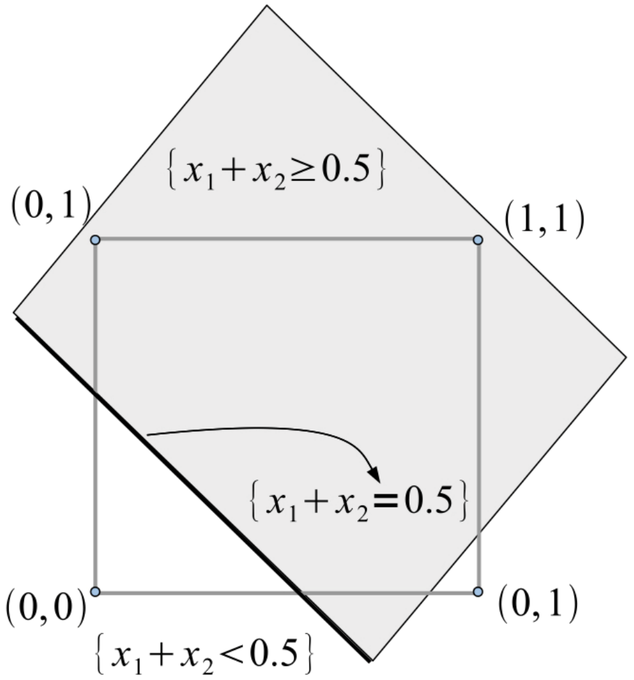
AND


−1
b=−1.5
x1
w1=1
w2=1
x2
y=x1∧x2


{x1+x2≥1.5}
(0,1)
(1,1)
{x1+x2=1.5}
{x1+x2<1.5}
(0,0)
(0,1)

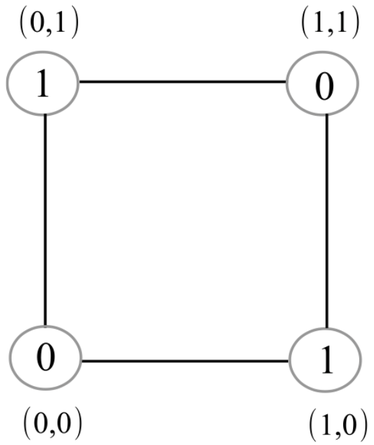
OR

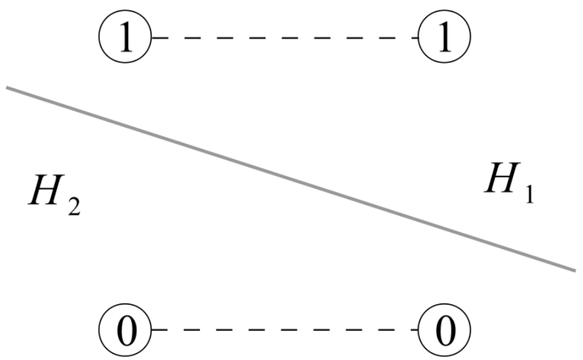

−1
b=−0.5
x1
w1=1
w2=1
x2
y=x1∨x2


{x1+x2≥0.5}
(0,1)
(1,1)
{x1+x2=0.5}
(0,0)
(0,1)
{x1+x2<0.5}

XOR Problem

Analytic Proof

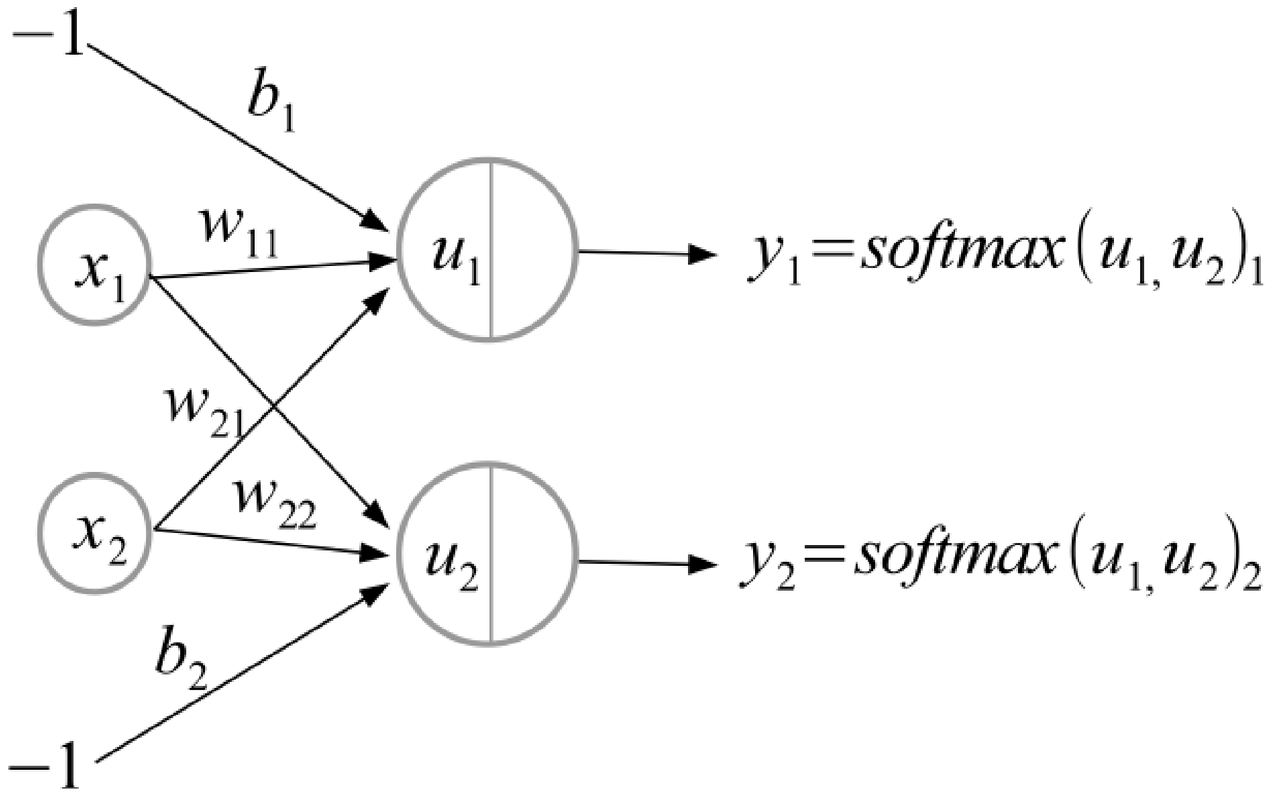




Algebraic Proof

- *Sigmoidal Neuron*
  - *Sigmoid Neuron*
  - *Scaled Logistic Neuron*
  - *Softsign Neuron*
- *Linear Neuron*
  - *Identity Neuron*
  - *ReLu Neuron*
- *AdaLine Neuron*
- *MadaLine Neuron*
- *Continuum Input Neuron*

Sigmoidal Neuron



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

$$\sigma_c(z) = \frac{1}{1+e^{-cz}}, \qquad c > 0$$

Tanh Neurons

Arctan Neurons

Softsign Neurons

Definition 2.2.1

$A\ function\ \sigma:\ \mathbb{R}\ \rightarrow [0,1]\ is\ called\ sigmoidal\ if$

$$\lim_{x\rightarrow-\infty}\sigma(x)=0 \quad \lim_{x\rightarrow\infty}\sigma(x)=1$$

$A\ measure\ \mu: a\ system\ of\ beliefs\ used\ to\ asses\ information$

$d\mu(x)=\mu(dx)\text{: } evaluation\ of\ the\ information\ cast\ into\ x$

$\int f(x)d\mu(x)\text{: } evalutaion\ of\ the\ function\ f(x)$
$under\ the\ system\ of\ beliefs\ \mu$

Definition 2.2.2

Let $\mu \in \mathcal{M}(I_n)$. A function σ is called discriminatory for the measure μ if :

$$\int \sigma(w^T x + \theta) d\mu(x) = 0 \quad for\ all\ w \in \mathbb{R}^n, \theta \in \mathbb{R} \quad \Rightarrow \quad \mu = 0$$

$$P_{w,\theta} = \{x; w^T x + \theta = 0\}$$
$$\mathcal{H}^+_{w,\theta} = \{x; w^T x + \theta > 0\}$$
$$\mathcal{H}^-_{w,\theta} = \{x; w^T x + \theta < 0\}$$

$$\mathbb{R}^n = \mathcal{H}^+_{w,\theta} \cup \mathcal{H}^-_{w,\theta} \cup P_{w,\theta}$$

Lemma 2.2.3

Let $\mu \in \mathcal{M}(I_n)$. If μ vanishes on all hyperplanes and open half − spaces in $\mathbb{R}^n$, then μ is zero. More precisely, if

$$\mu\left(P_{w,\theta}\right) = 0 \quad \mu\left(\mathcal{H}^{+}_{w,\theta}\right) = 0 \quad \text{for all } w \in \mathbb{R}^n, \theta \in \mathbb{R}$$

then $\mu = 0$.

Proposition 2.2.4

Any continuous sigmoidal function is discriminatory for all measures $\mu \in \mathcal{M}(I_n)$.

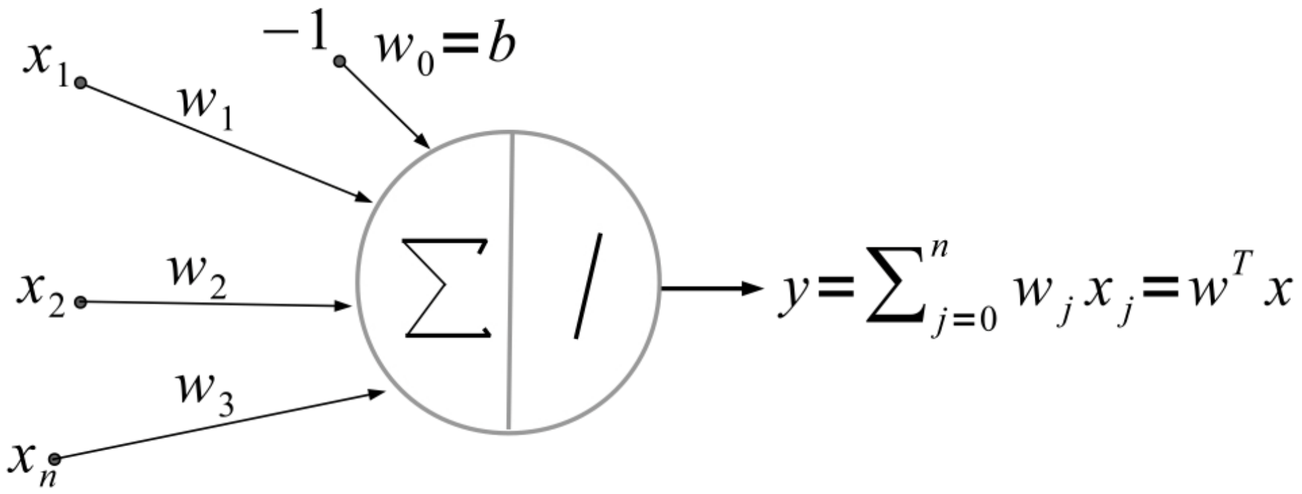
$$dy = \sum_i^n \frac{\partial y}{\partial w_i} dw_i + \frac{\partial y}{\partial b} db =$$

$$\sum_i^n \varphi'(w^T x - b) x_i dw_i + \varphi'(w^T x - b)\,(-1) db =$$

$$\varphi'(w^T x - b)(\sum_i^n x_i dw_i \; - db)$$

$$\nabla y = (\nabla_w y, \partial_b y) = \left(x\varphi'(w^T x - b), -\varphi'(w^T x - b)\right)$$
$$= \varphi'(w^T x - b)(x, -1)$$

Linear Neuron



$$Y = \sum_j^n w_j X_j = w^T X = X^T w$$

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathbb{E}[(Z - Y)^2]$$

$$\begin{aligned}
\mathbb{E}[(Z-Y)^2] &= \mathbb{E}[Z^2 - 2ZY + Y^2] = \mathbb{E}[Z^2 - 2ZX^T\mathbf{w} + (\mathbf{w}^T X)(X^T\mathbf{w})] \\
&= \mathbb{E}[Z^2] - 2\mathbb{E}[ZX^T]\mathbf{w} + \mathbf{w}^T\mathbb{E}[XX^T]\mathbf{w} \\
&= c - 2\mathbf{b}^T\mathbf{w} + \mathbf{w}^T A\mathbf{w}.
\end{aligned}$$

$$\mathbf{b} = \mathbb{E}[ZX]$$

$$c = \mathbb{E}[Z^2]$$

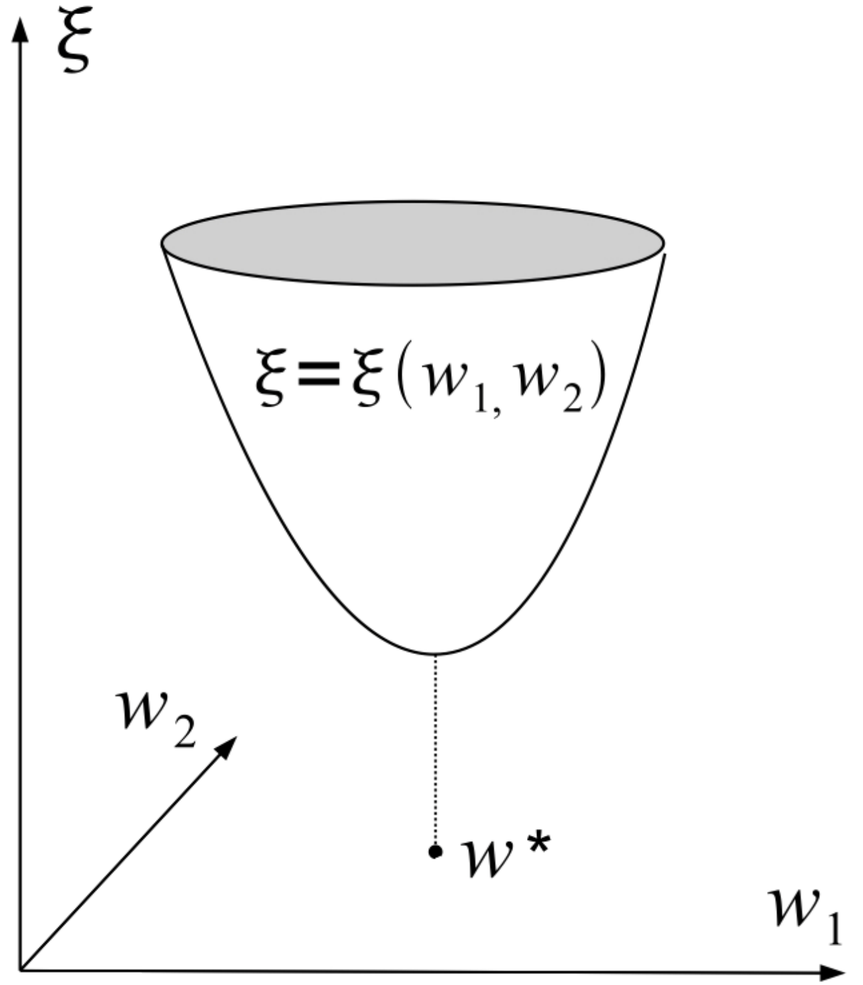
$$A = \mathbb{E}[XX^T] = \begin{pmatrix} \mathbb{E}[X_0X_0] & \mathbb{E}[X_0X_1] & \cdots & \mathbb{E}[X_0X_n] \\ \mathbb{E}[X_1X_0] & \mathbb{E}[X_1X_1] & \cdots & \mathbb{E}[X_1X_n] \\ \cdots & \cdots & \cdots & \cdots \\ \mathbb{E}[X_nX_0] & \mathbb{E}[X_nX_1] & \cdots & \mathbb{E}[X_nX_n] \end{pmatrix}$$

Error Function : $\xi(\mathbf{w}) = c - 2\mathbf{b}^T\mathbf{w} + \mathbf{w}^T A\mathbf{w}$

$$\nabla_{\mathbf{w}}\xi(\mathbf{w}) = 2A\mathbf{w} - 2\mathbf{b}$$

$$H_\xi(\mathbf{w}) = 2A$$

$$\mathbf{w}^* = A^{-1}\mathbf{b}$$



$$
\begin{aligned}
\mathbf{w}^{(j+1)} &= \mathbf{w}^{(j)} - \eta \nabla_{\mathbf{w}} \xi(\mathbf{w}^{(j)}) \\
&= \mathbf{w}^{(j)} - 2\eta(A\mathbf{w}^{(j)} - \mathbf{b}) \\
&= (\mathbb{I}_n - 2\eta A)\mathbf{w}^{(j)} + 2\eta\mathbf{b} \\
&= M\mathbf{w}^{(j)} + 2\eta\mathbf{b},
\end{aligned}
$$

$$M = \mathbb{I}_n - 2\eta A$$

$$
\begin{aligned}
\mathbf{w}^{(j)} &= M^j\mathbf{w}^{(0)} + (M^{j-1} + M^{j-2} + \cdots + M + \mathbb{I}_n)\, 2\eta\mathbf{b} \\
&= M^j\mathbf{w}^{(0)} + (\mathbb{I}_n - M^j)(\mathbb{I}_n - M)^{-1}\, 2\eta\mathbf{b} \\
&= M^j\mathbf{w}^{(0)} + (\mathbb{I}_n - M^j)A^{-1}\mathbf{b}.
\end{aligned}
$$

$$\lim_{j\to\infty} M^j = \mathbb{O}_n$$

Step 1. Show that $\lambda_i < 1$.
Let λ_i denote an eigenvalue of M. Then $\det(M - \lambda_i \mathbb{I}_n) = 0$. Substituting for M, this becomes $\det\left(A - \frac{1-\lambda_i}{2\eta}\mathbb{I}_n\right) = 0$. This implies that $\alpha_i = \frac{1-\lambda_i}{2\eta}$ is an eigenvalue of matrix A. Since A is positive definite and nondegenerate, it follows that $\alpha_i > 0$, which implies that $\lambda_i < 1$.

Step 2. Show that $\lambda_i > 0$ for η small enough.
The condition $\lambda_i > 0$ is equivalent to $\frac{1-\lambda_i}{\alpha_i} < \frac{1}{\alpha_i}$, where we used that A has positive eigenvalues. This can be written in terms of η as $2\eta < \frac{1}{\alpha_i}$. Hence, the learning rate has to be chosen such that

$$0 < \eta < \min_i \frac{1}{2\alpha_i}. \tag{5.5.4}$$

The closed-form expression (5.5.3) is not of much practical use, since it contains the inverse A^{-1}. In real life we use the iterative formula

$$\mathbf{w}^{(j+1)} = M\mathbf{w}^{(j)} + 2\eta\mathbf{b},$$

where the learning rate η satisfies the inequality (5.5.4).

Error Estimation

Error

$$\begin{aligned}\mathbf{w}^{(j)} - \mathbf{w}^* &= M^j\mathbf{w}^{(0)} + (\mathbb{I}_n - M^j)\mathbf{w}^* - \mathbf{w}^* \\ &= M^j(\mathbf{w}^{(0)} - \mathbf{w}^*).\end{aligned}$$

$$\epsilon_j = |\mathbf{w}^{(j)} - \mathbf{w}^*| = |M^j(\mathbf{w}^{(0)} - \mathbf{w}^*)|$$

$$\leq \|M\|^j|(\mathbf{w}^{(0)} - \mathbf{w}^*)| = \mu^j d$$

$$\big|\big|M\big|\big| = \max_i \lambda_i \qquad \lambda_i < 1$$

Empirical Estimation

$$\xi(w) = \mathbb{E}[(Z - Y)^2]$$

$$\widehat{\xi}(w) = \frac{1}{m}\sum_{j=1}^{m}(z^{(j)} - \mathbf{w}^T\mathbf{x}^{(j)})^2 = \frac{1}{m}\sum_{j=1}^{m}\epsilon_j^2$$

$$\epsilon_j = z^{(j)} - \mathbf{w}^T\mathbf{x}^{(j)}$$

$$\begin{aligned}
\nabla_w\widehat{\xi}(w) &= \nabla_{\mathbf{w}}\epsilon_j^2 = 2\epsilon_j\partial_{\mathbf{w}}\epsilon_j = 2\epsilon_j\partial_{\mathbf{w}}(z^{(j)} - \mathbf{w}^T\mathbf{x}^{(j)}) \\
&= -2\epsilon_j\mathbf{x}^{(j)}. \\
\mathbf{w}^{(j+1)} &= \mathbf{w}^{(j)} - \eta\nabla_{\mathbf{w}}\widehat{\xi} \\
&= \mathbf{w}^{(j)} - \eta(-2\epsilon_j\mathbf{x}^{(j)}) = \mathbf{w}^{(j)} + 2\eta\epsilon_j\mathbf{x}^{(j)}
\end{aligned}$$

$$\mathbf{w}^{(j+1)} = \mathbf{w}^{(j)} + 2\eta(z^{(j)} - \mathbf{w}^T\mathbf{x}^{(j)})\mathbf{x}^{(j)}$$

Continuum Input Neuron

The weight associated with the value x is given by w(dx).

$$w \text{ is a weighting measure on } [0,1]$$

$$y = \sigma\left(\int_0^1 x dw(x) \right)$$

If w represents the distribution measure of X,
then the neuron output depends on its expectation, $y = \sigma(E[X])$.

Dirac Measure

$$\delta_{x_0}(A) = \begin{cases} 1, & \text{if } x_0 \in A \\ 0, & \text{if } x_0 \notin A \end{cases}$$

$$y = \sigma\Big(\int_0^1 x d\delta_{x_0}(x) \Big) = \sigma(x_0)$$

Discrete Measure

$$\mu(A) = \sum_{x_i \in E} w_i \delta_{x_i}(A)$$

$$\forall A \in \mathcal{B}([0,1])$$

$$y = \sigma\Big(\int_0^1 x \, d\mu(x) \Big)$$

$$= \sigma\Big(\sum_{j=1}^{n} w_j x_j \Big)$$

Lebesgue Measure

Radon-Nikodym Theorem

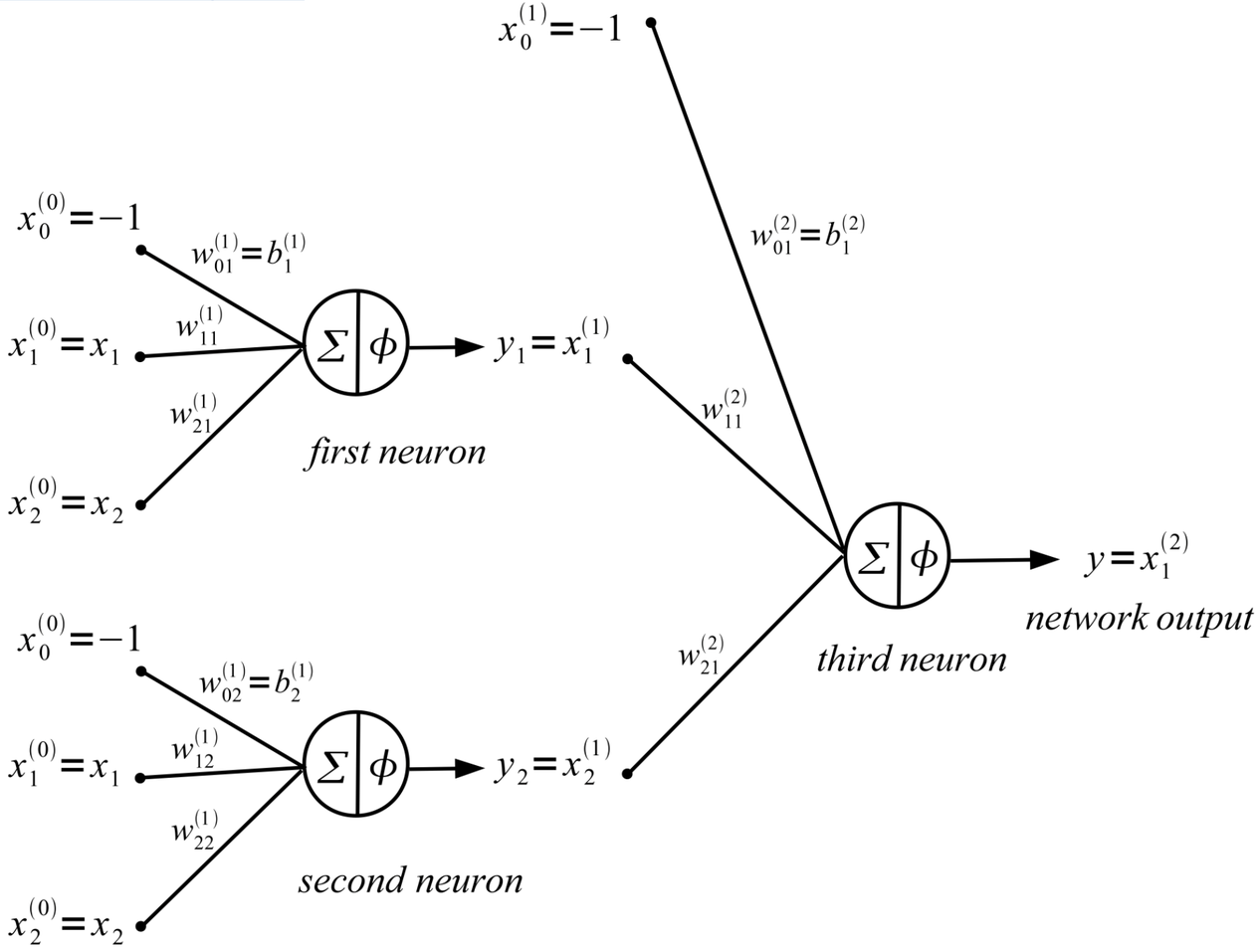
$$d\mu(x) = p(x)dx$$

$$y = \sigma\Big(\int_0^1 x \, d\mu(x) \Big)$$

$$= \sigma\Big(\int_0^1 x p(x) \, dx \Big)$$

CH6.

Deep Learning Architecture



$$s_1^{(1)} = w_{01}^{(1)} x_0^{(0)} + w_{11}^{(1)} x_1^{(0)} + w_{21}^{(1)} x_2^{(0)} = \sum_{i=1}^{2} w_{i1}^{(1)} x_i^{(0)} - b_1^{(1)}$$

$$s_2^{(1)} = w_{02}^{(1)} x_0^{(0)} + w_{12}^{(1)} x_1^{(0)} + w_{22}^{(1)} x_2^{(0)} = \sum_{i=1}^{2} w_{i2}^{(1)} x_i^{(0)} - b_2^{(1)}$$

$$y_1 = x_1^{(1)} = \phi(s_1^{(1)}) = \phi\Big(\sum_{i=1}^{2} w_{i1}^{(1)} x_i^{(0)} - b_1^{(1)} \Big)$$

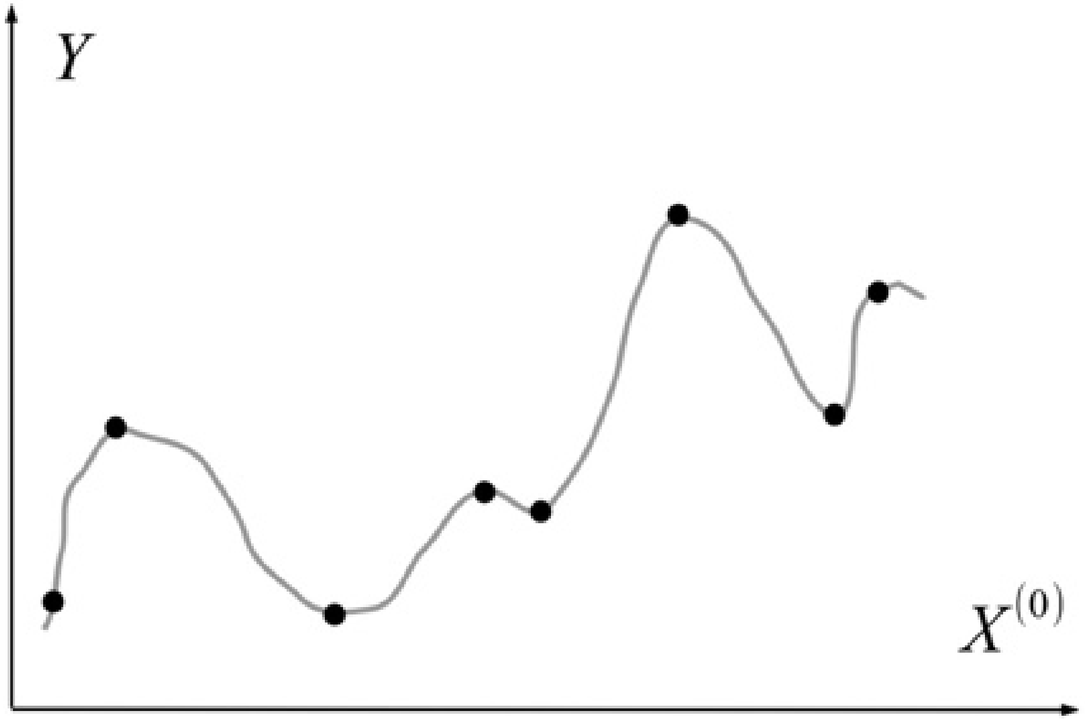
$$y_2 = x_2^{(1)} = \phi(s_2^{(1)}) = \phi\Big(\sum_{i=1}^{2} w_{i2}^{(1)} x_i^{(0)} - b_2^{(1)} \Big)$$

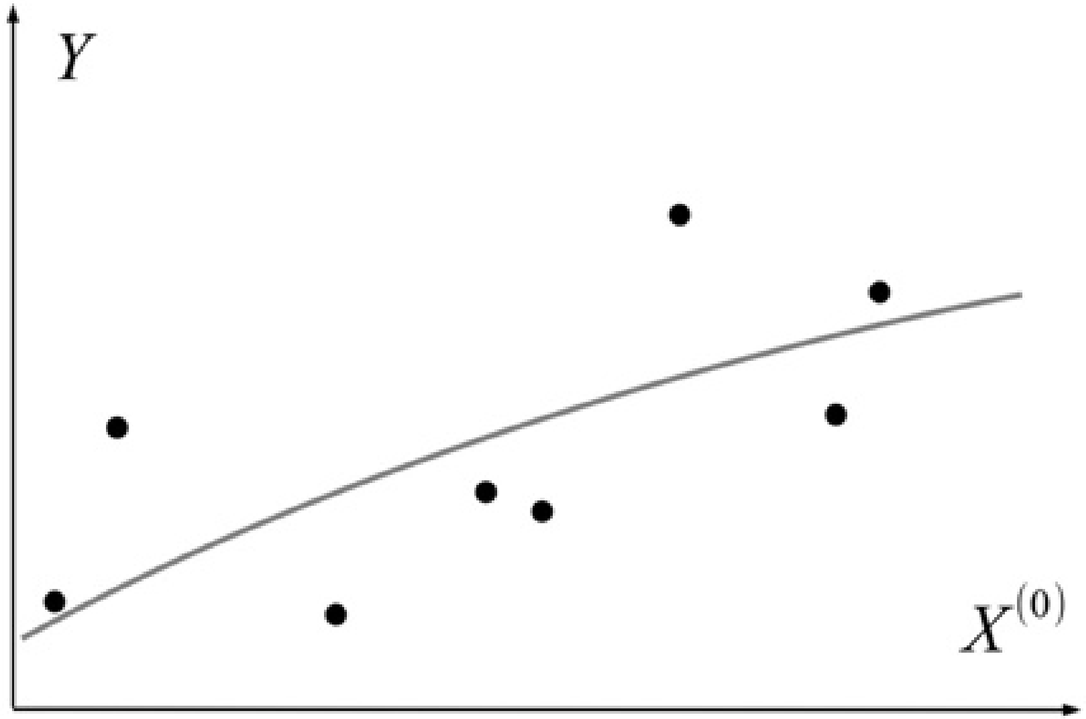
$$\begin{pmatrix} s_1^{(1)} \\ s_2^{(1)} \end{pmatrix} = \begin{pmatrix} w_{11}^{(1)} & w_{21}^{(1)} \\ w_{12}^{(1)} & w_{22}^{(1)} \end{pmatrix} \begin{pmatrix} x_1^{(0)} \\ x_2^{(0)} \end{pmatrix} - \begin{pmatrix} b_1^{(1)} \\ b_2^{(1)} \end{pmatrix}$$

$$X^{(1)} = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} x_1^{(1)} \\ x_2^{(1)} \end{pmatrix} = \begin{pmatrix} \phi(s_1^{(1)}) \\ \phi(s_2^{(1)}) \end{pmatrix} = \phi(s^{(1)})$$

$$f_{w,b}(X) = \phi\Big(W^{(2)^T} \phi\big(W^{(1)^T} X - b^{(1)} \big) - b^{(2)} \Big)$$

Total Variance



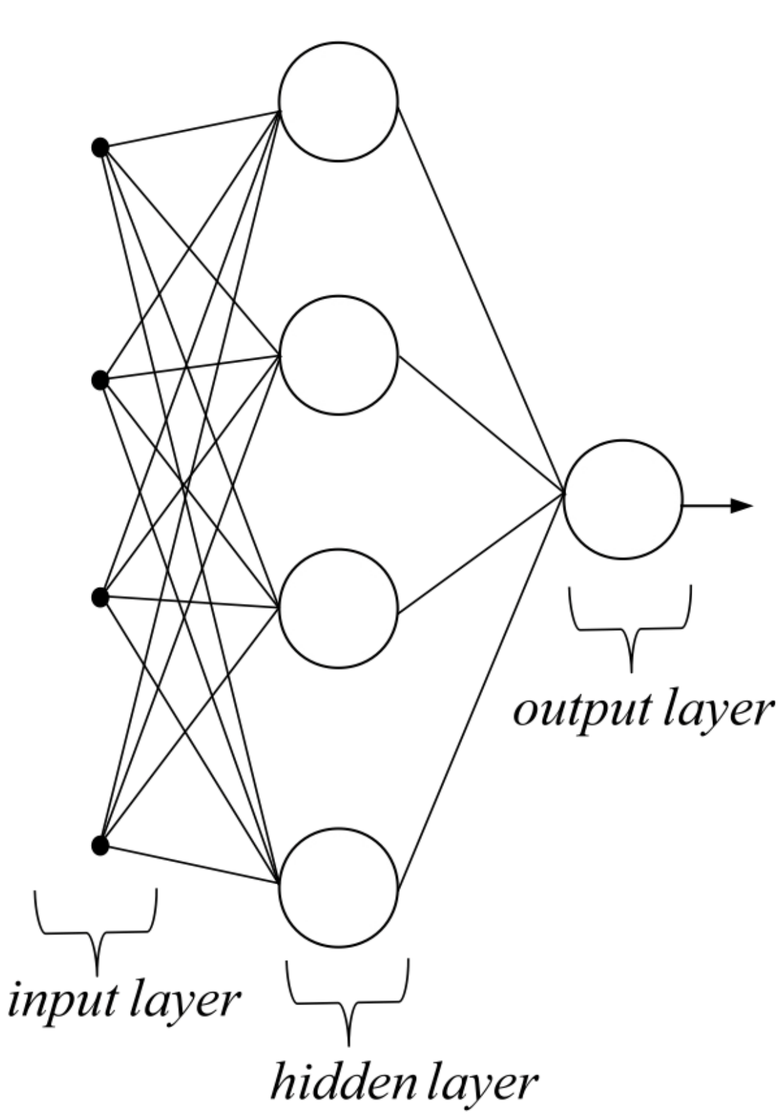


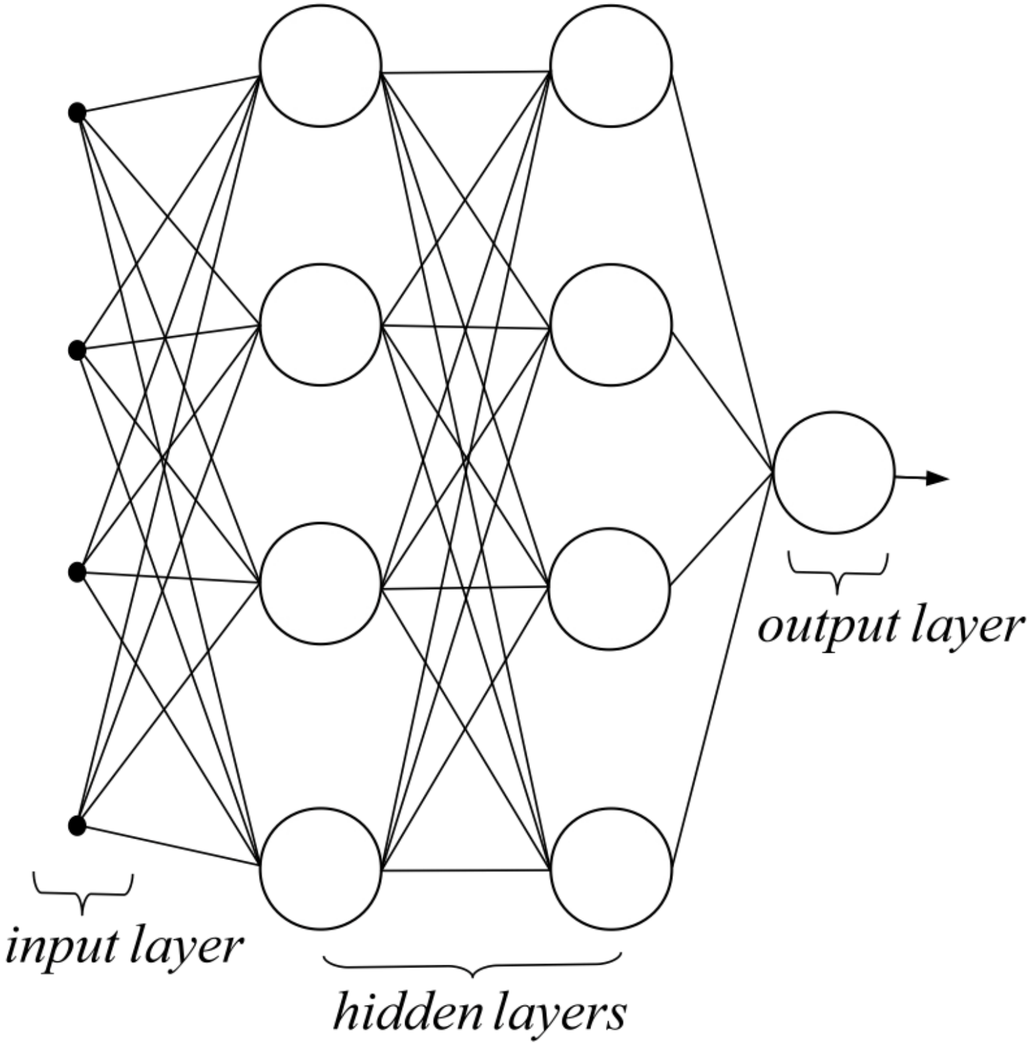
Total Variance

$$V(f) = \int |df(X)|$$

$$dY = \phi'(s^{(2)})W^{(2)^T}\phi'(s^{(1)})W^{(1)^T}\, dX^{(0)}$$

$$\|W^{(\ell)}\|_1 \leq 1, \text{ or } \|W^{(\ell)}\|_2 \leq 1$$


output layer
input layer
hidden layer


output layer
input layer
hidden layers

BackPropagation

$$\nabla C = (\nabla_w C, \nabla_b C)$$

$$\delta_j^{(\ell)} = \frac{\partial C}{\partial s_j^{(\ell)}}$$

the sensitivity of the error with respect to the signal $s_j^{(l)}$

$$\frac{\partial C}{\partial b_1^{(1)}} = \frac{\partial C}{\partial w_{01}^{(1)}} = \frac{\partial C}{\partial s_1^{(1)}} \frac{\partial s_1^{(1)}}{\partial w_{01}^{(1)}} = \delta_1^{(1)} x_0^{(0)} = -\delta_1^{(1)}$$

$$\frac{\partial C}{\partial w_{11}^{(1)}} = \frac{\partial C}{\partial s_1^{(1)}} \frac{\partial s_1^{(1)}}{\partial w_{11}^{(1)}} = \delta_1^{(1)} x_1^{(0)}$$

$$\frac{\partial C}{\partial w_{21}^{(1)}} = \frac{\partial C}{\partial s_1^{(1)}} \frac{\partial s_1^{(1)}}{\partial w_{21}^{(1)}} = \delta_1^{(1)} x_2^{(0)}$$

BackPropagation

$$\nabla C = (\nabla_w C, \nabla_b C)$$

$$\delta_j^{(\ell)} = \frac{\partial C}{\partial s_j^{(\ell)}}$$

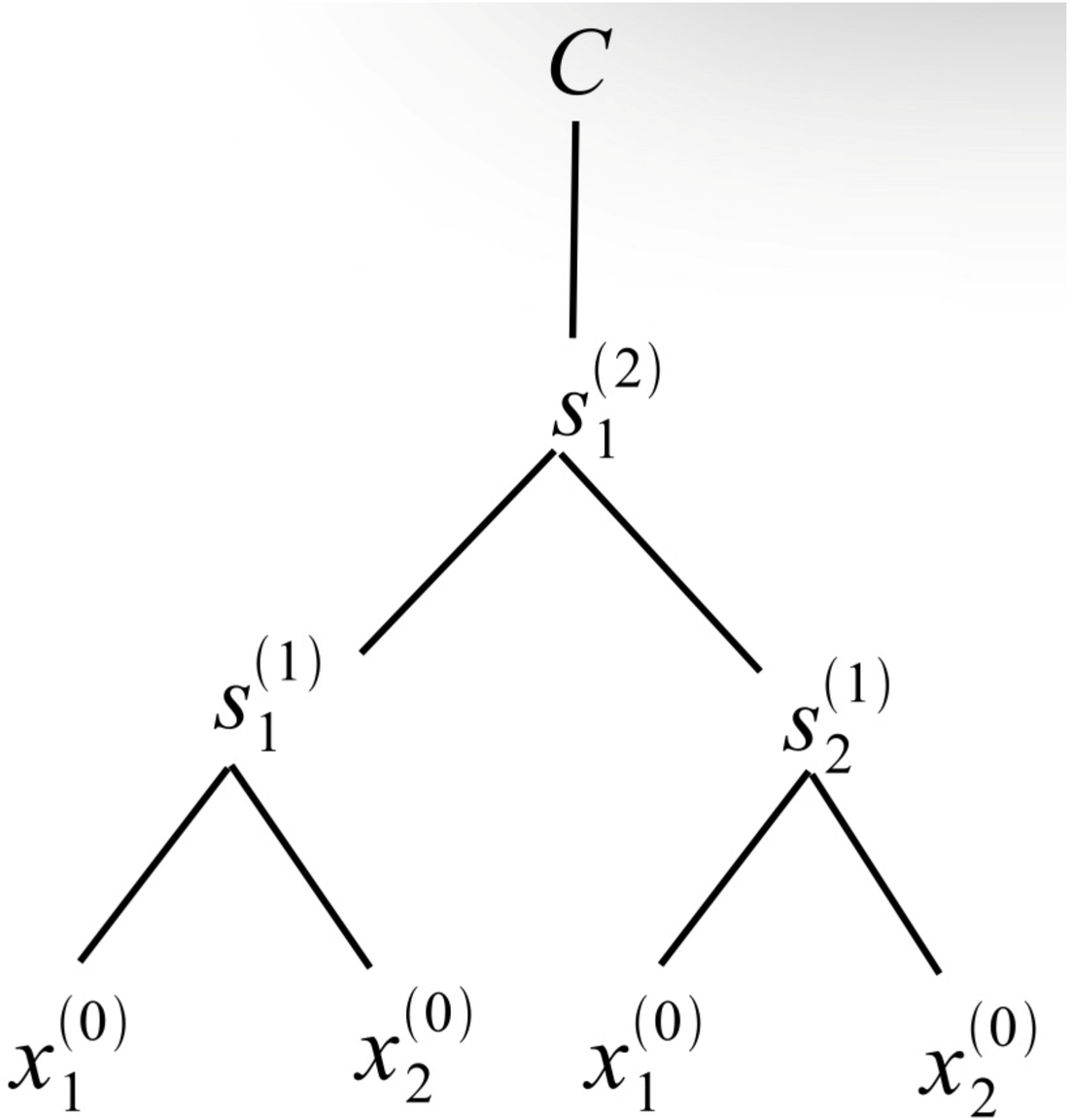
the sensitivity of the error with respect to the signal $s_j^{(l)}$

$$\frac{\partial C}{\partial b_1^{(2)}} = \frac{\partial C}{\partial w_{01}^{(2)}} = \frac{\partial C}{\partial s_1^{(2)}} \frac{\partial s_1^{(2)}}{\partial w_{01}^{(2)}} = \delta_1^{(2)} x_0^{(1)} = -\delta_1^{(2)}$$

$$\frac{\partial C}{\partial w_{11}^{(2)}} = \frac{\partial C}{\partial s_1^{(2)}} \frac{\partial s_1^{(2)}}{\partial w_{11}^{(2)}} = \delta_1^{(2)} x_1^{(1)}$$

$$\frac{\partial C}{\partial w_{21}^{(2)}} = \frac{\partial C}{\partial s_1^{(2)}} \frac{\partial s_1^{(2)}}{\partial w_{21}^{(2)}} = \delta_1^{(2)} x_2^{(1)},$$

BackPropagation



BackPropagation

$$\nabla C = (\nabla_w C, \nabla_b C)$$

$$\delta_j^{(\ell)} = \frac{\partial C}{\partial s_j^{(\ell)}}$$

the sensitivity of the error with respect to the signal $s_j^{(l)}$

Generalization

$$\frac{\partial C}{\partial b_j^{(\ell)}} = -\delta_j^{(\ell)}$$

$$\frac{\partial C}{\partial w_{ij}^{(\ell)}} = \delta_j^{(\ell)} x_i^{(\ell-1)}$$

BackPropagation

$$b_j^{(\ell)}(k+1) = b_j^{(\ell)}(k) - \eta \frac{\partial C}{\partial b_j^{(\ell)}}\left(w_{ij}^{(\ell)}(k), b_j^{(\ell)}(k)\right)$$

$$w_{ij}^{(\ell)}(k+1) = w_{ij}^{(\ell)}(k) - \eta \frac{\partial C}{\partial w_{ij}^{(\ell)}}\left(w_{ij}^{(\ell)}(k), b_j^{(\ell)}(k)\right)$$

$$b_j^{(\ell)}(k+1) = b_j^{(\ell)}(k) + \eta \delta_j^{(\ell)}$$

$$w_{ij}^{(\ell)}(k+1) = w_{ij}^{(\ell)}(k) - \eta \delta_j^{(\ell)} x_i^{(\ell-1)}(k)$$

$$\nabla C = (\nabla_w C, \nabla_b C) = \delta_j^{(\ell)}(x_i^{(\ell-1)}, -1)$$

BackPropagation

$$\delta_1^{(1)} = \frac{\partial C}{\partial s_1^{(1)}} = \frac{\partial C}{\partial s_1^{(2)}} \frac{\partial s_1^{(2)}}{\partial s_1^{(1)}} = \delta_1^{(2)} \frac{\partial s_1^{(2)}}{\partial s_1^{(1)}}$$

$$s_1^{(2)} = -w_{01}^{(2)} + w_{11}^{(2)} \phi(s_1^{(1)}) + w_{21}^{(2)} \phi(s_2^{(1)})$$

$$\frac{\partial s_1^{(2)}}{\partial s_1^{(1)}} = w_{11}^{(2)} \phi'(s_1^{(1)})$$

$$\delta_1^{(1)} = \delta_1^{(2)} w_{11}^{(2)} \phi'(s_1^{(1)})$$

BackPropagation

$$
\begin{aligned}
\delta_2^{(1)} &= \frac{\partial C}{\partial s_2^{(1)}} = \frac{\partial C}{\partial s_1^{(2)}} \frac{\partial s_1^{(2)}}{\partial s_2^{(1)}} \\
&= \delta_1^{(2)} \frac{\partial}{\partial s_2^{(1)}} \Big(- w_{01}^{(2)} + w_{11}^{(2)} \phi(s_1^{(1)}) + w_{21}^{(2)} \phi(s_2^{(1)}) \Big) \\
&= \delta_1^{(2)} w_{21}^{(2)} \phi'(s_2^{(1)}).
\end{aligned}
$$

$$
\begin{pmatrix} \delta_1^{(1)} \\ \\ \delta_2^{(1)} \end{pmatrix} = \delta_1^{(2)} \begin{pmatrix} w_{11}^{(2)} \phi'(s_1^{(1)}) \\ \\ w_{21}^{(2)} \phi'(s_2^{(1)}) \end{pmatrix}
$$

BackPropagation

$$\delta_1^{(2)} = \frac{\partial C}{\partial s_1^{(2)}} = \frac{\partial C}{\partial Y}\frac{\partial Y}{\partial s_1^{(2)}} = \frac{\partial C}{\partial Y}\frac{\partial}{\partial s_1^{(2)}}\phi(s_1^{(2)}) = \frac{\partial C}{\partial Y}\phi'(s_1^{(2)})$$

$$\begin{pmatrix} \delta_1^{(1)} \\ \\ \delta_2^{(1)} \end{pmatrix} = \frac{\partial C}{\partial Y}\phi'(s_1^{(2)}) \begin{pmatrix} w_{11}^{(2)}\phi'(s_1^{(1)}) \\ \\ w_{21}^{(2)}\phi'(s_2^{(1)}) \end{pmatrix}$$

BackPropagation (Generalization)

Assumption : $$C(w,b) = \frac{1}{2}\sum_{j=1}^{d^{(L)}}(x_j^{(L)} - z_j)^2 = \frac{1}{2}\|x^{(L)} - z\|^2$$

$$\delta_j^{(L)} = \frac{\partial C}{\partial s_j^{(L)}} = (x_j^{(L)} - z_j)\phi'(s_j^{(L)})$$

$$\delta_i^{(\ell-1)} = \frac{\partial C}{\partial s_i^{(\ell-1)}} = \sum_{j=1}^{d^{(\ell)}} \frac{\partial C}{\partial s_j^{(\ell)}} \frac{\partial s_j^{(\ell)}}{\partial s_i^{(\ell-1)}} = \sum_{j=1}^{d^{(\ell)}} \delta_j^{(\ell)} \frac{\partial s_j^{(\ell)}}{\partial s_i^{(\ell-1)}}$$

BackPropagation (Generalization)

$$\frac{\partial s_j^{(\ell)}}{\partial s_i^{(\ell-1)}} = \frac{\partial}{\partial s_i^{(\ell-1)}}\Big(\sum_{i=1}^{d^{(\ell-1)}} w_{ij}^{(\ell)} x_i^{(\ell-1)} - b_j^{(\ell)}\Big)$$

$$= \frac{\partial}{\partial s_i^{(\ell-1)}}\Big(\sum_{i=1}^{d^{(\ell-1)}} w_{ij}^{(\ell)} \phi(s_i^{(\ell-1)}) - b_j^{(\ell)}\Big)$$

$$= w_{ij}^{(\ell)} \phi'(s_i^{(\ell-1)}).$$

BackPropagation Formula

$$\delta_i^{(\ell-1)} = \phi'(s_i^{(\ell-1)}) \sum_{j=1}^{d^{(\ell)}} \delta_j^{(\ell)} w_{ij}^{(\ell)}$$

Master Equations

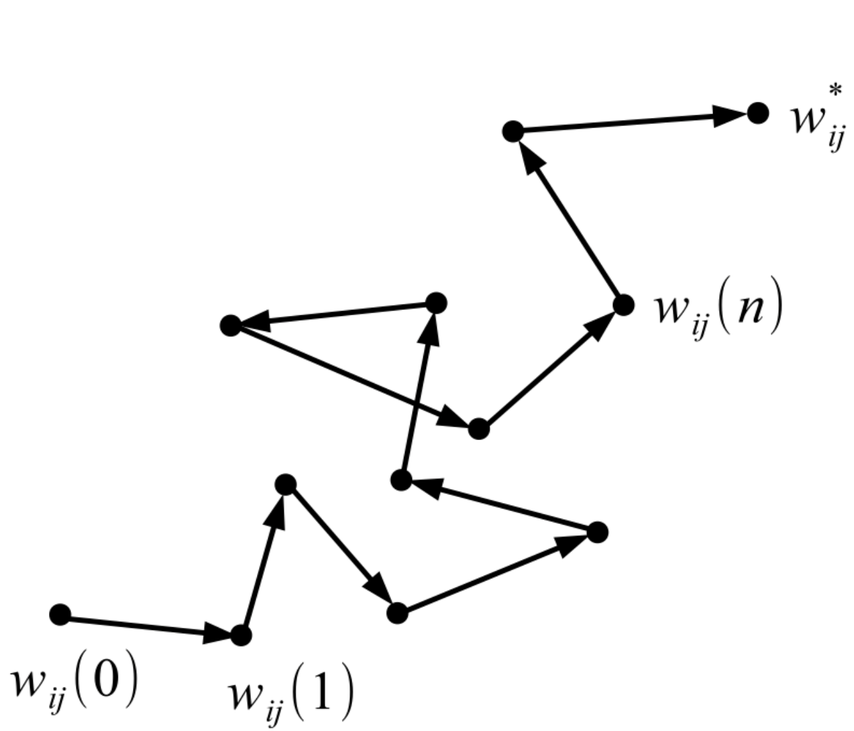
$$
\begin{aligned}
x_j^{(\ell)} &= \phi\Big(\sum_{i=1}^{d^{(\ell-1)}} w_{ij}^{(\ell)} x_i^{(\ell-1)} - b_j^{(\ell)}\Big), \qquad 1 \le j \le d^{(\ell)} \\
\delta_j^{(L)} &= (x_j^{(L)} - z_j)\phi'(s_j^{(L)}), \qquad 1 \le j \le d^{(L)} \\
\delta_i^{(\ell-1)} &= \phi'(s_i^{(\ell-1)}) \sum_{j=1}^{d^{(\ell)}} \delta_j^{(\ell)} w_{ij}^{(\ell)}, \qquad 1 \le i \le d^{(\ell-1)} \\
\frac{\partial C}{\partial w_{ij}^{(\ell)}} &= \delta_j^{(\ell)} x_i^{(\ell-1)} \\
\frac{\partial C}{\partial b_j^{(\ell)}} &= -\delta_j^{(\ell)}.
\end{aligned}
$$

$$
\begin{aligned}
X^{(\ell)} &= \phi\Big(W^{(\ell)^T} X^{(\ell-1)} - B^{(\ell)}\Big) \\
\delta^{(L)} &= (x^{(L)} - z) \odot \phi'(s^{(L)}) \\
\delta^{(\ell-1)} &= \big(W^{(\ell)}\delta^{(\ell)}\big) \odot \phi'(s^{(\ell-1)}) \\
\frac{\partial C}{\partial W^{(\ell)}} &= X^{(\ell-1)}\delta^{(\ell)^T} \\
\frac{\partial C}{\partial B^{(\ell)}} &= -\delta^{(\ell)}.
\end{aligned}
$$

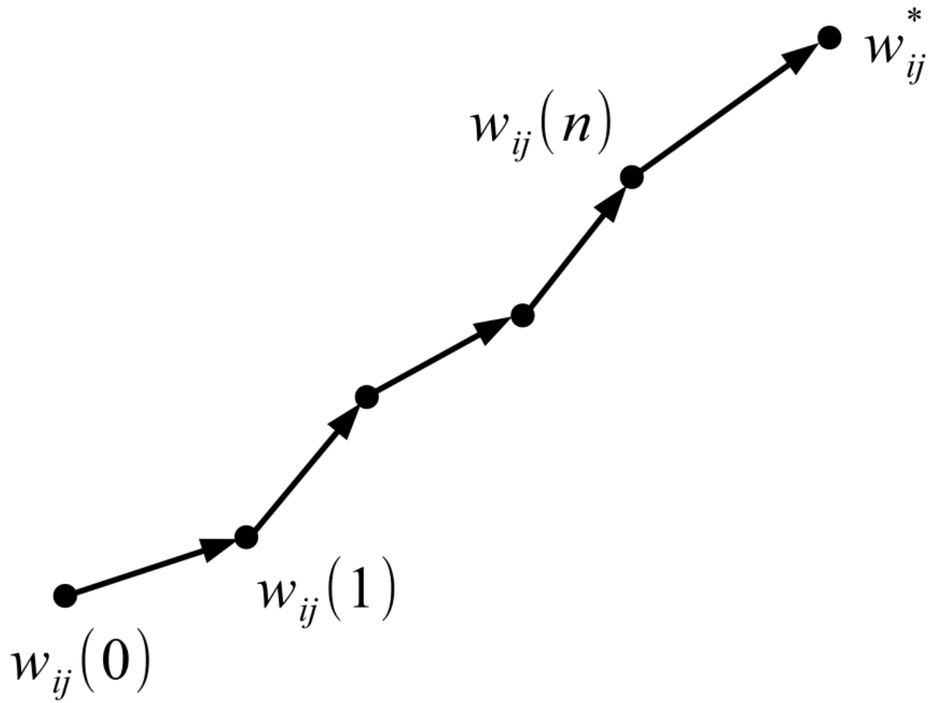
Vanishing Gradient Problem

Cost formula	Deltas
$\frac{1}{2}\|x^{(L)} - z\|^2$	$\delta_j^{(L)} = (x_j^{(L)} - z_j)\sigma'(s_j^{(L)})$
$-\sum z_k \ln x_k^{(L)}$	$\delta_j^{(L)} = -z_j(1 - \sigma(s_j^{(L)}))$
$-\sum z_k \ln x_k^{(L)} - (1 - z_k)\ln(1 - x_k^{(L)})$	$\delta_j^{(L)} = x_j^{(L)} - z_j$

Batch Training



a



b

Batch Training

$$\widetilde{\nabla C} = \frac{1}{N} \sum_{k=1}^{N} \nabla C(X^{(0,k)})$$

By Central Limit Theorem,
“Averages tend to have a smaller variance
than each individual outcome”

FeedForward Network

Definition 6.2.5 *Let* $U_\ell = \{1, 2, \ldots, d^{(\ell)}\}$, $0 \leq \ell \leq L$, *and consider the sequence of affine functions* $\alpha_1, \ldots, \alpha_L$

$$\alpha_\ell : \mathcal{F}(U_{\ell-1}) \to \mathcal{F}(U_\ell)$$

and the sequence of activation functions $\phi^{(\ell)} : \mathbb{R} \to \mathbb{R}$. *Then the corresponding feedforward neural network is the sequence of maps* $f_0, f_1, \ldots, f_L$, *where*

$$f_\ell = \phi^{(\ell)} \circ \alpha_\ell \circ f_{\ell-1}, \quad 1 \leq \ell \leq L,$$

with f_0 *given.*

$$f_L = \phi^{(L)} \circ \alpha_L \circ \phi^{(L-1)} \circ \alpha_{L-1} \circ \cdots \circ \phi^{(1)} \circ \alpha_1$$

Weight Initialization

Background

(i) If the weights are too close to zero, then the variance of the input signal decreases as it passes through each layer of the network.

$$X_j^{(\ell)} = \phi\Big(\sum_i W_{ij}^{(\ell)} X_i^{(\ell-1)} - b_j^{(\ell)}\Big)$$

$$Var(f(X)) \approx f'\big(\mathbb{E}[X]\big)^2 Var(X)$$

Weight Initialization

Background

(i) If the weights are too close to zero, then the variance of the input signal decreases as it passes through each layer of the network.

$$Var(X_j^{(\ell)}) \approx \phi'\Big(\sum_{\cdot} W_{ij}^{(\ell)} \mathbb{E}[X_i^{(\ell-1)}] - b_j^{(\ell)}\Big)^2 \Big(\sum_{\cdot} W_{ij}^{(\ell)} Var(X_i^{(\ell-1)})\Big)$$

$$\sum_i W_{ij}^{(\ell)} Var(X_i^{(\ell-1)}) \leq \Big(\sum_i (W_{ij}^{(\ell)})^2\Big)^{1/2} \Big(\sum_i Var(X_i^{(\ell-1)})^2\Big)^{1/2}$$

$$\sum_j Var(X_j^{(\ell)})^2 < C^2 \sum_{i,j} (W_{ij}^{(\ell)})^2 \sum_i Var(X_i^{(\ell-1)})^2$$

Weight Initialization

Background

(ii) If the weights are too large, then either the variance of the signal tends to get amplified as it passes through the network layers, or the network approaches a vanishing gradient problem.

$$Var(X_j^{(\ell)}) = \sum_i W_{ij}^{(\ell)} Var(X_i^{(\ell-1)})$$

Weight Initialization

Notation : $$Y_j = \phi\Big(\sum_i W_{ij}X_i + b_j\Big)$$

$$\begin{aligned} Var(Y_j) &= \phi'\Big(\sum_i \mathbb{E}[W_{ij}X_i] + b_j\Big)^2 Var\Big(\sum_i W_{ij}X_i\Big) \\ &= \phi'(b_j)^2 \sum_i Var(W_{ij}X_i) \\ &= \phi'(b_j)^2 \sum_i Var(W_{ij})Var(X_i) \\ &= N\phi'(0)^2 Var(W_{ij})Var(X_i), \end{aligned}$$

$$\mathbb{E}[W_{ij}X_i] = \mathbb{E}[W_{ij}]\mathbb{E}[X_i] = 0$$

$$Var(Y_j) = Var(X_i)$$

Weight Initialization

$$Var(W_{ij}) = \frac{1}{N\phi'(0)^2}$$

Assumption :
“Normal”

$$W_{ij} \sim \mathcal{N}\left(0, \frac{1}{N\phi'(0)^2}\right)$$

Assumption :
“Uniform”
on [-a, a]

$$W_{ij} \sim Unif\left[-\frac{\sqrt{3}}{\phi'(0)\sqrt{N}}, \frac{\sqrt{3}}{\phi'(0)\sqrt{N}}\right]$$

Xavier Initialization

$$Var\left[\frac{\partial C}{\partial W_{ij}^{(\ell-1)}}\right] = Var\left[\frac{\partial C}{\partial W_{ij}^{(\ell)}}\right]$$

$$Var\left[\delta_j^{(\ell-1)} X_i^{(\ell-2)}\right] = Var\left[\delta_j^{(\ell)} X_i^{(\ell-1)}\right]$$

$$\delta_i^{(\ell-1)} = \phi'(s_i^{(\ell-1)}) \sum_{j=1}^{N'} \delta_j^{(\ell)} W_{ij}^{(\ell)} = \sum_{j=1}^{N'} \delta_j^{(\ell)} W_{ij}^{(\ell)}$$

$$N' = d^{(\ell)}$$

Xavier Initialization

$$\mathbb{E}\Big[\sum_{j=1}^{N'} \delta_j^{(\ell)} W_{ij}^{(\ell)}\Big] = \sum_{j=1}^{N'} \mathbb{E}[\delta_j^{(\ell)}]\mathbb{E}[W_{ij}^{(\ell)}] = 0$$

$$\mathbb{E}[W_{ij}^{(\ell)}] = 0$$

$$Var\big[\delta_j^{(\ell-1)}\big] Var\big[X_i^{(\ell-2)}\big] = Var\big[\delta_j^{(\ell)}\big] Var\big[X_i^{(\ell-1)}\big]$$

$$Var\big[\delta_j^{(\ell-1)}\big] = Var\big[\delta_j^{(\ell)}\big]$$

$$Var(\delta_i^{(\ell-1)}) = \sum_{j=1}^{N'} Var(\delta_j^{(\ell)}) Var(W_{ij}^{(\ell)}) = N' Var(\delta_j^{(\ell)}) Var(W_{ij}^{(\ell)})$$

Weight Initialization

$$Var(W_{ij}^{(\ell)}) = \frac{1}{N}$$

Assumption :
“Normal”

$$W_{ij} \sim \mathcal{N}\left(0, \frac{2}{N+N'}\right)$$

Assumption :
“Uniform” with zero mean

$$Unif\left[-\frac{\sqrt{6}}{\sqrt{N+N'}}, \frac{\sqrt{6}}{\sqrt{N+N'}}\right]$$

Strong & Weak Prior

$$H(p) = -\int p(x) \ln p(x)\, dx$$

Strong Prior : Low Entropy

Weak Prior : High Entropy

$$W_{ij} \sim Unif\Big[-\frac{\sqrt{3}}{\phi'(0)\sqrt{N}}, \frac{\sqrt{3}}{\phi'(0)\sqrt{N}}\Big]$$

$$H(p_W) = \ln \frac{2\sqrt{3}}{\phi'(0)\sqrt{N}}$$

$$W_{ij} \sim \mathcal{N}\Big(0, \frac{2}{N+N'}\Big)$$

$$H(p_W) = \ln \frac{2\sqrt{e}}{\sqrt{N+N'}}$$

Homework

다음 중 선택하여, 최소 1문제(Exercise 1문제) 이상 해주세요!

1.Computation

2.Application

3.Interpretation

4.Googling & Interpretation & Application

Homework

1. Computation

Exercise 4.17.2 Consider the quadratic function $Q(x) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - b\mathbf{x}$, with A nonsingular square matrix of order n.

(a) Find the gradient ∇Q;

(b) Write the gradient descent iteration;

(c) Find the Hessian H_Q;

(d) Write the iteration given by Newton's formula and compute its limit.

Exercise 6.6.10 Let $p(x)$ be the uniform distribution over the interval $[a, b]$. Show that $H(p) = \ln(b - a)$.

Exercise 6.6.11 Let $p(x)$ be the one-dimensional normal distribution with mean μ and standard deviation σ. Show that its entropy is $H(p) = \ln(\sigma\sqrt{2\pi e})$.

Homework

2. Application

Example 4.2.1 Consider the function $f : (0,1) \times (-2,2) \to \mathbb{R}$, given by $f(x,y) = \frac{1}{2}y^2 - x$. Its graph has a canyon-type shape, with the minimum at the point $(1,0)$, see Fig. 4.8. Let (x^0, y^0) be a fixed point in the function domain. Since the gradient is given by $\nabla f(x,y)^T = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}) = (-1, y)$, the equation (4.2.4) writes as

$$\begin{aligned} x^{n+1} &= x^n + \delta \\ y^{n+1} &= (1-\delta)y^n. \end{aligned}$$

Example 4.13.1 We shall consider the one-dimensional case given by the objective function $f(x) = \dfrac{1}{2}x^2$, $x \in (a,b)$, with $a > 0$. The diffusion $dX_t = b(X_t)dt + \sigma(X_t)dW_t$, $X_0 = x_0 \in (a,b)$ is one-dimensional, with $b(x)$ and $\sigma(x)$ continuous functions. Since the Hessian is $f''(x) = 1$, the equation $\sigma^2 = \lambda \frac{1}{f''(x)}$ implies $\sigma = \sqrt{\lambda}$. Also, $b(x) = -\eta f'(x) = -\eta x$. Since $L = \inf_{a<x<b} f'(x)^2 = a^2$, then $\lambda = 2a^2\eta$. The stochastic differential equation becomes the *Langevin's equation*

Homework

3. Interpretation

Exercise 5.10.4 Explain the equivalence between the linear regression algorithm and the learning of a linear neuron.

Homework

4. Googling & Interpretation & Application

Batch Training Section에서, **Central Limit Theorem**에 따라서,
각각의 training을 하는 것보다, Batch training이
분산이 더 작게 나온다는 점에 대해 고민해보아요!

THANK YOU