Final Review - ch5

Monday, 7 December 2020 3:27 PM

Ch5 Optimization

Problem statement:

$$\min_{x \in \mathbb{R}^n} f(x)$$

unconstrained optimization

constrained optimization

$$\min_{x \in \mathbb{R}^n} f(x)$$
$$\text{s.t. } g_i(x) \le 0 \quad i = 1, 2, \cdots, p$$
$$h_i(x) = 0 \quad i = 1, 2, \cdots, q.$$

Section 5.1 Smooth Unconstrained Optimization
Problem Statement:

Consider unconstrained optimization

$$\min_{x \in \mathbb{R}^n} f(x). \qquad (OPT)$$

We assume $f(x)$ is differentiable.

We say $x^{(*)}$ is a solution of (OPT) if

$$f(x^{(*)}) \le f(x) \qquad \forall\, x \in \mathbb{R}^n.$$

而呢個$x^{(*)}$未必存在的，需要f 滿足兩個條件：

Theorem: If f is

① continuous (i.e., $f(x^{(n)}) \to f(x)$ as $x^{(n)} \to x$)

② Coecive. (i.e., $f(x^{(n)}) \to +\infty$ as $\|x^{(n)}\| \to +\infty$)

then there exists at least one solution of (OPT)

Pf26. Proving if f is continuous and coercive, there exists at least one solution of unconstrained smooth optimization

注意：呢個theorem的reverse未必hold true

Optimization solution的特性：

那什麼時候$x^{(*)}$才是一個solution呢?

1. Zero-th order condition:

$$f(x^{(*)}) \leq f(x) \quad \forall\, x \in \mathbb{R}^n.$$

2. First order condition

Theorem: Assume f is differentiable at $x^{(*)}$. Then

$$x^{(*)} = \arg\min_{x \in \mathbb{R}^n} f(x) \Longrightarrow \nabla f(x^{(*)}) = 0$$

Pf27. proof of first order condition in unconstrained smooth optimization problem

注意呢個thoerem的reverse版本不一定hold true

即是，如果某個點是minimum,他的gradient是0

但是gradient是0不一定代表就能找到minimum

gradient是0的時候可能出現的情況；

From this example, we see that $x^{(*)}$ with $\nabla f(x^{(*)}) = 0$ can be

- global minimizer, i.e., $x^{(*)} = \arg\min_{x \in \mathbb{R}^n} f(x)$ (See $x^{(1)}$)
- local minimizer, i.e. (See $x^{(3)}$)

  $\exists \varepsilon > 0$ s.t. $f(x^{(*)}) \leq f(x)$ $\forall\, x$: $\|x - x^{(*)}\| \leq \varepsilon$.
- global maximizer, i.e.,

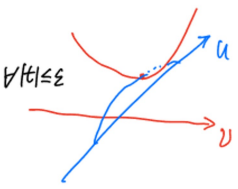
  $f(x^{(*)}) \geq f(x)$ $\forall\, x \in \mathbb{R}^n$ (See $x^{(5)}$)
- local maximizer, i.e.,

  $\exists \varepsilon > 0$, s.t. $f(x^{(*)}) \geq f(x)$ $\forall\, x$: $\|x - x^{(*)}\| \leq \varepsilon$ (See $x^{(2)}$)
- Saddle points (for $x \in \mathbb{R}^n$ with $n \geq 2$)

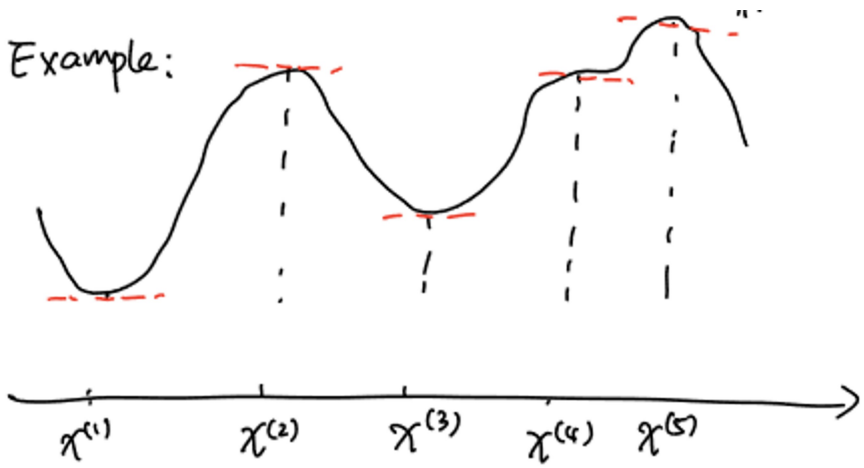
  $\exists\, u, v \in \mathbb{R}^n$, s.t. $f(x^{(*)}) \geq f(x^{(*)} + tu)$

  $\varepsilon > 0$ $f(x^{(*)}) \leq f(x^{(*)} + tv)$

  $\forall |t| \leq \varepsilon$

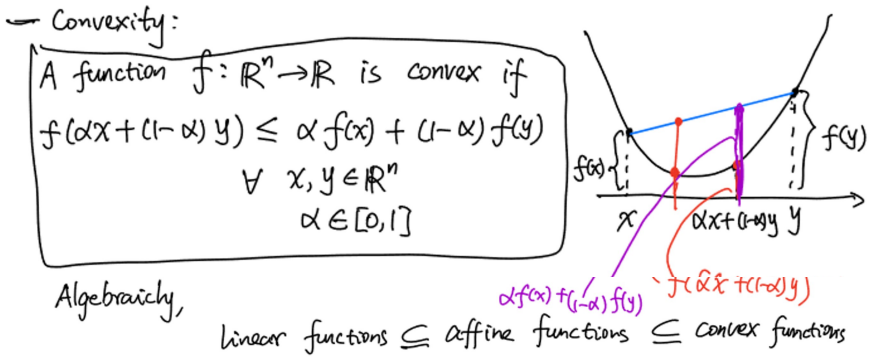


- others (See $x^{(4)}$)



Sufficient condition for optimality

那現在我們想知道，在什麼情況下grad = 0就肯定找到了minimum



Example for convex function:

1. F(x) =x^2
2. $||x||^2$
3. $||x||$
4. Any affine function
5. Linear combination of affine functions, the constants need to be non-negative numbers:

Let $f_1, f_2, \cdots, f_m$ be convex function.

Let $c_1, c_2, \cdots, c_m$ be non-negative numbers

then $f = \sum_{i=1}^{m} c_i f_i$ is convex

6. F 。 g is convex if f and g are convex
7. F(Ax+b) if A is matrix, b is vector

Theorem: Convex + differentiable, grad = 0 <==> argmin

Theorem: If $f: \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable, then

$x^{(*)} = \arg\min_{x \in \mathbb{R}^n} f(x) \iff \nabla f(x^{(*)}) = 0$

pf28: proof of the theorem: Convex + differentiable, grad = 0 <==> argmin

Strictly convex:

Strictly convex:

Strictly convex function:

A function $f: \mathbb{R}^n \to \mathbb{R}$ is strictly convex if

$$f(\alpha x + (1-\alpha) y) < \alpha f(x) + (1-\alpha) f(y)$$

Examples for strictly convex function:

$F(x) = x^2$, $||x||^2$

Examples for convex function:

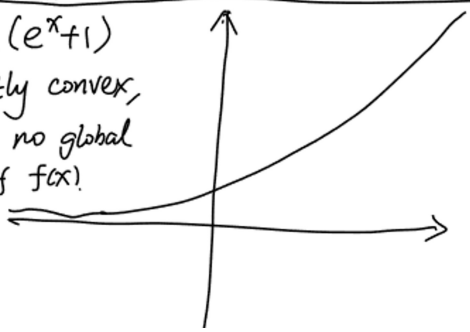
$F(x) = |x|$, $|x|_1$

Theorem: If a function is strictly convex, the solution is unique if it exists

Pf29

Remark: Even if f is convex/strictly convex, the existence of a solution of $\min_{x \in \mathbb{R}^n} f(x)$ is NOT guaranteed.

Example: $f(x) = \ln(e^x + 1)$ is strictly convex, but there is no global minimizer of $f(x)$.



Gradient Descent

We want to find an algorithm to solve $\min_{x \in \mathbb{R}^n} f(x)$

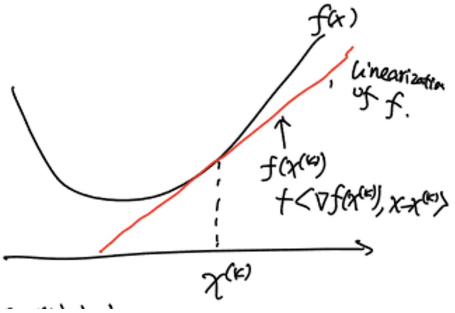
We find the minimizer iteratively

Let $x^{(k)}$ be the current estimation of a minimizer.

Instead of minimizing f itself, we minimize an approximation of f. (linearization)

$$f(x) \approx f(x^{(k)}) + \langle \nabla f(x^{(k)}), x - x^{(k)} \rangle$$

We have several problems:



This approximation is accurate only when x is close to $x^{(k)}$.

Therefore, instead of $\min_{x \in \mathbb{R}^n} f(x)$, we minimize the linear approximation and the distance of x to $x^{(k)}$ simultaneously

Standard gradient descent

To achive this, we solve

$$x^{(k+1)} = \arg\min_{x \in \mathbb{R}^n} \underbrace{f(x^{(k)}) + \langle \nabla f(x^{(k)}), x - x^{(k)} \rangle}_{\text{to make the linear approximation small}} + \underbrace{\frac{1}{2\alpha_k} \|x - x^{(k)}\|_2^2}_{\text{to make } x \text{ close to } x^{(k)}}$$

呢到都可以用1 norm

Penalize term

where $\alpha_k > 0$ is a parameter to balance the two terms.

Let $F(x) = f(x^{(k)}) + \langle \nabla f(x^{(k)}), x - x^{(k)} \rangle + \frac{1}{2\alpha_k} \|x - x^{(k)}\|_2^2$

Pf30 It can be checked F(x) is continuous, coercive, strictly convex

$\nabla f(x^{(k+1)}) =$ It can be checked that $x^{(k+1)} = x^{(k)} - \alpha_k \nabla f(x^{(k)})$

This is gradient descent algorithm (Pf31)

About alpha:

1. Exact line search:

$$\alpha_k = \arg\min_{\alpha \geq 0} f\left(x^{(k)} - \alpha \nabla f(x^{(k)})\right).$$

1. ② Linear search by back-tracking:
   - Try a very large α_k.
   - Test whether or not α_k is good enough:
     - — Yes. Go to the next iteration.
     - — No. Decrease α_k and try again.

Example:

Example: Armijo-Goldstein backtracking

Choose a large α_0. Choose $\beta < 1$ (e.g. $\beta = 0.9$).

At step k:

Discount rate

count

$\alpha_k = \alpha_{k-1}$

$x^{(k+1)} = x^{(k)} - \alpha_k \nabla f(x^{(k)})$

if $f(x^{(k)}) - f(x^{(k+1)}) < \frac{\alpha_k}{2} \|\nabla f(x^{(k)})\|_2^2$

then $\alpha_k = \beta \alpha_k$ and goto

otherwise go to the next iteration

There are many other creteria to test a good step size

Search "Armijo-Goldstein Condition" in google

③ Use a fixed step size.

因爲 $\nabla f(x) = 0$ 就有可能 minimize 了.

After some proof, Choice of alpha should be in (0,1/M), where

$\langle \nabla f(x) - \nabla f(y), x-y \rangle \le M \|x-y\|_2^2 \quad \forall\, x, y \in \mathbb{R}^n$

(The gradient $\nabla f(x)$ doesn't change too much w.r.t x)

(Pf31)

Summary theorem of above properties for gradient descent

Theorem: Assume ① f is differentiable and $\exists$ a solution of $\min_{x \in \mathbb{R}^n} f(x)$.

② $\langle \nabla f(x) - \nabla f(y), x-y \rangle \le M \|x-y\|_2^2, \quad \forall x, y \in \mathbb{R}^n$.

③ $\alpha \in (0, \frac{2}{M})$ Reasonable step size

Then, the sequence $\{x^{(k)}\}$ generated by

$x^{(k+1)} = x^{(k)} - \alpha \nabla f(x^{(k)})$ and $x^{(0)} \in \mathbb{R}^n$

Satisfies: i). $f(x^{(k+1)}) < f(x^{(k)})$ (the function value decreases)

ii) $\lim_{k \to +\infty} \|\nabla f(x^{(k)})\|_2 = 0$ (the limit has a vanishing gradient)

Section 5.2. Case Studies of Gradient Descent

Section 5.2.1 Least Squares

Linear Regression Problem Statement:

via minimizing the squares error

$$\min_{\substack{a \in \mathbb{R}^n \\ b \in \mathbb{R}}} \frac{1}{2} \sum_{i=1}^{N} (\langle x_i, a \rangle + b - y_i)^2$$

Let $X = \begin{bmatrix} x_1^T & 1 \\ \vdots & \vdots \\ x_N^T & 1 \end{bmatrix} \in \mathbb{R}^{N \times (n+1)} \quad \beta = \begin{bmatrix} a \\ b \end{bmatrix} \in \mathbb{R}^{n+1} \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} \in \mathbb{R}^N$.

Then we need to solve the least squares (LS) problem

$$\min_{\beta \in \mathbb{R}^{n+1}} \frac{1}{2} \|X\beta - y\|_2^2$$

- We consider the standard LS problem

$$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - b\|_2^2, \quad \text{where } A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m.$$

Let $f(x) = \frac{1}{2} \|Ax - b\|_2^2$.

It can be checked that f(x) is convex, differentiable
Pf32

Normal Equation of the least squares problem:

• Therefore,

$$x^{(*)} = \arg\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax-b\|_2^2 \iff A^T(Ax^{(*)}-b)=0$$

$$\iff \boxed{A^TA\,x^{(*)} = A^Tb}$$

Geometric Interpretation for normal equation

• Geometric Interpretation

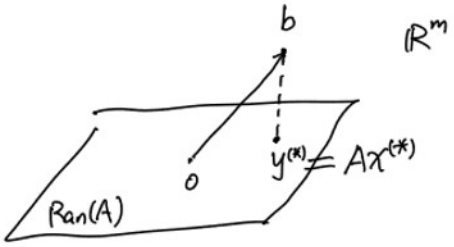
$A\in\mathbb{R}^{m\times n}$

$$x^{(*)} = \arg\min_{x\in\mathbb{R}^n} \frac{1}{2}\|Ax-b\|_2^2$$

$$\Updownarrow\ y=Ax$$

$$\begin{cases} y^{(*)} = \arg\min_{y\in Ran(A)} \frac{1}{2}\|y-b\|_2^2 \\ y^{(*)} = Ax^{(*)} \end{cases}$$

$$\Updownarrow$$



$$b-Ax^{(*)} \perp Ran(A) \iff \langle b-Ax^{(*)}, Az\rangle = 0 \quad \forall z\in\mathbb{R}^n$$

$$\iff \langle A^T(b-Ax^{(*)}), z\rangle = 0 \quad \forall z\in\mathbb{R}^n$$

$$\iff A^T(b-Ax^{(*)}) = 0$$

$$\iff A^TAx^{(*)} = A^Tb.$$

Gradient descent for LS:

• If A^TA is invertible, then $\frac{1}{2}\|Ax-b\|_2^2$ is strictly convex.
So, the least squares has a unique solution.

• To find $x^{(*)}$, we may use a gradient descent:

$$x^{(k+1)} = x^{(k)} - \alpha_k A^T(Ax^{(k)}-b), \quad k=0,1,2,\dots$$

$$\alpha_k = \arg\min_{\alpha\in\mathbb{R}} f\left(x^{(k)} - \alpha A^T(Ax^{(k)}-b)\right)$$

$$\alpha_k = \arg\min_{\alpha \in \mathbb{R}} \underbrace{f\left(x^{(k)} - \alpha A^T(Ax^{(k)} - b)\right)}_{g(\alpha)}$$

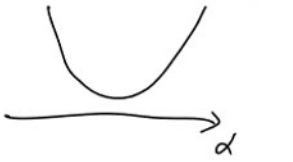
Steepest descent:

Define $g(\alpha)$.

— $g(\alpha)$ is convex, differentiable, such that

$$g'(\alpha_k) = 0 \iff \alpha_k = \frac{\|A^T(Ax^{(k)} - b)\|_2^2}{\|AA^T(Ax^{(k)} - b)\|_2^2}$$

(Actually, $g(\alpha)$ is a quadratic function of α)



So the full algorithm is

$$\begin{cases} g^{(k)} = A^T(Ax^{(k)} - b) \\ \alpha_k = \dfrac{\|g^{(k)}\|_2^2}{\|Ag^{(k)}\|_2^2} \\ x^{(k+1)} = x^{(k)} - \alpha_k g^{(k)} \end{cases}$$

(Called Steepest Descent)

Regularization in LS:

When A^TA is not invertible, we have many solutions of LS problem

Ridge Regression

- Ridge regression.

$$\min_{x \in \mathbb{R}^n} \underbrace{\frac{1}{2}\|Ax - b\|_2^2 + \frac{\lambda}{2}\|x\|_2^2}_{f(x)}, \quad \lambda > 0$$

① $f(x)$

— f is strictly convex, because $\begin{cases} \frac{1}{2}\|Ax-b\|_2^2 \text{ is convex} \\ \frac{\lambda}{2}\|x\|_2^2 \text{ is strictly convex.} \end{cases}$

② — f is differentiable and

$$\nabla f(x) = A^T(Ax - b) + \lambda x$$

— $x^{(*)} = \arg\min_{x \in \mathbb{R}^n} \frac{1}{2}\|Ax - b\|_2^2 + \frac{\lambda}{2}\|x\|_2^2$

$$\Updownarrow$$

$$\nabla f(x^{(*)}) = 0$$

$$\Updownarrow$$

$$A^T(Ax^{(*)} - b) + \lambda x^{(*)} = 0 \iff \boxed{(A^TA + \lambda I)x^{(*)} = A^Tb}$$ ③

— For any $\lambda > 0$, $A^TA + \lambda I$ is always invertible.

Because $A^TA + \lambda I$ is always SPD: ④

– For any $\lambda > 0$, $A^TA + \lambda I$ is always invertible, (4)

because $A^TA + \lambda I$ is always SPD:

① $(A^TA + \lambda I)^T = A^TA + \lambda I$

② $x^T(A^TA + \lambda I)x = x^TA^TAx + \lambda x^Tx$
$= \|Ax\|_2^2 + \lambda \|x\|_2^2 > 0$ if $x \neq 0$.

– We can use steepest descent to find $x^{(*)}$.

① + ② → ③ + ④

Kernel Ridge Regression:

choose a kernel function $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ s.t.

$K(x, z) = \langle \phi(x), \phi(z) \rangle \quad \forall\, x, z \in \mathbb{R}^n$.

$$\min_{c \in \mathbb{R}^N} \sum_{i=1}^{N} \left(\sum_{j=1}^{N} c_j K(x_i, x_j) - y_i \right)^2 + \lambda \sum_{i=1}^{N} \sum_{j=1}^{N} c_i c_j K(x_i, x_j)$$

$\Updownarrow$ Let $K = [K(x_i, x_j)]_{i,j}$

$$\min_{c \in \mathbb{R}^N} \frac{1}{2} \|Kc - y\|_2^2 + \frac{\lambda}{2} c^T K c$$

$\Updownarrow$

$c^{(*)}$ is a solution $\iff$ $K^T(Kc^{(*)} - y) + \lambda K c^{(*)} = 0$

$\iff$ $(K^TK + \lambda K)c^{(*)} = K^Ty$.

Ex: $f(x) = \frac{1}{2} x^TAx$

LASSO cannot use gradient descent to solve

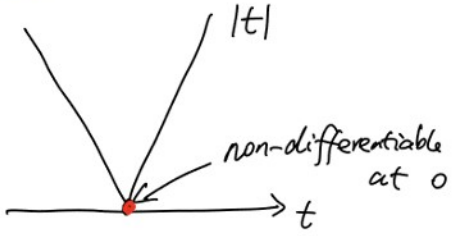
- LASSO regression.

$$\min_{x \in \mathbb{R}^n} \underbrace{\frac{1}{2}\|Ax - b\|_2^2 + \lambda \|x\|_1}_{f(x)}, \qquad \lambda > 0$$

– f is convex, because both $\frac{1}{2}\|Ax-b\|_2^2$ and $\lambda\|x\|_1$ are convex.

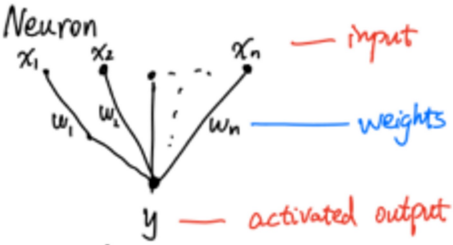
– f is NOT differentiable, so that the gradient descent is NOT applicable.

$\|x\|_1 = \sum_{i=1}^{n} |x_i|$



Section 5.2.2 Neural Network Training

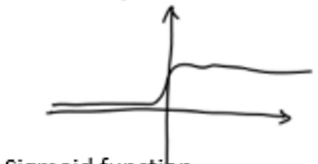
- Artificial Neuron



$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i\right) \quad \text{or} \quad y = \sigma(\langle w, x \rangle)$$

activation

Activation function $\sigma : \mathbb{R} \to \mathbb{R}$



Sigmoid function

classical choice

ReLU (popular recently)

. . . many others

Let $w^{(j)} \in \mathbb{R}^n$ be the weights in neuron j, $j = 1, \cdots, p$.

Denote $W = [w^{(1)} \cdots w^{(p)}] \in \mathbb{R}^{n \times p}$

Let $c = \begin{bmatrix} c_1 \\ \vdots \\ c_p \end{bmatrix} \in \mathbb{R}^p$ be the weights in the output layer.

Then the input-output relation is

$$y = \sum_{j=1}^{p} c_j \sigma(\langle w^{(j)}, x \rangle)$$

$$= \langle c, \sigma(W^T x) \rangle \equiv f_{W,c}(x), \quad \text{where } x \in \mathbb{R}^n.$$

We may use the least squares error:

$$\min_{\substack{W \in \mathbb{R}^{n \times p} \\ c \in \mathbb{R}^p}} \sum_{i=1}^{m} \left(f_{W,c}(x^{(i)}) - y_i \right)^2$$

— Let $F_i(W, c) = \left(f_{W,c}(x^{(i)}) - y_i \right)^2$

and $F(W, c) = \sum_{i=1}^{m} F_i(W, c)$.

— Then $F, F_i : \mathbb{R}^{n \times p} \times \mathbb{R}^p \left(\equiv \mathbb{R}^{(n+1)p} \right) \to \mathbb{R}$

發現即使function是non-linear, 都可以用gradient descent得到好的神經網絡

We need to compute grad(F(w,p))

Pf 34: gradient of least squares function in neural network is:

Pf 34: gradient of least squares function in neural network is:

$$\frac{\partial F}{\partial w^{(j)}}(W,c) = \sum_{i=1}^{m} 2\left(f_{W,c}(x^{(i)}) - y_i\right)\left(\frac{\partial}{\partial w^{(j)}} f_{W,c}(x^{(i)})\right)$$

$$\frac{\partial}{\partial w^{(j)}} f_{W,c}(x^{(i)}) = \nabla(g_4 \circ g_3)(w^{(j)}) = g_4'(g_3(w^{(j)})) \cdot \nabla g_3(w^{(j)})$$

$$= c_j \cdot \sigma'(\langle w^{(j)}, x^{(i)} \rangle) \cdot x^{(i)}$$

$$\frac{\partial F}{\partial w^{(j)}}(W,c) = \sum_{i=1}^{m} 2\left(f_{W,c}(x^{(i)}) - y_i\right) \cdot c_j \cdot \sigma'(\langle w^{(j)}, x^{(i)} \rangle) \cdot x^{(i)}$$

$$= 2c_j \sum_{i=1}^{m} \left[\left(f_{W,c}(x^{(i)}) - y_i\right) \sigma'(\langle w^{(i)}, x^{(i)} \rangle)\right] x^{(i)}$$

$$\frac{\partial F}{\partial c}(W,c) = \frac{\partial}{\partial c} \sum_{i=1}^{m} \left(f_{W,c}(x^{(i)}) - y_i\right)^2$$

$$\frac{\partial}{\partial c} f_{W,c}(x^{(i)}) = \sigma(W^T x^{(i)})$$

$$\frac{\partial F}{\partial c}(W,c) = \sum_{i=1}^{m} 2\left(f_{W,c}(x^{(i)}) - y_i\right) \sigma(W^T x^{(i)})$$

- So, GD for neural network training is:

$$\begin{cases} w^{(j,k+1)} = w^{(j,k)} - \alpha_k 2 c_j^{(k)} \sum_{i=1}^{m} \left[\left(f_{w^{(k)},c^{(k)}}(x^{(i)}) - y_i \right) \sigma'\left(\langle w^{(j,k)}, x^{(i)} \rangle \right) \right] x^{(i)} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j = 1, \cdots, p. \\ c^{(k+1)} = c^{(k)} - \alpha_k \sum_{i=1}^{m} 2 \left(f_{w^{(k)},c^{(k)}}(x^{(i)}) - y_i \right) \sigma\left((w^{(k)})^T x^{(i)} \right) \end{cases}$$

GD計算成本低:

In the computation, all the computations are simple, except:

— $f_{w^{(k)},c^{(k)}}(x^{(i)})$ (output of the neural network)

— $(w^{(k)})^T x^{(i)}$ (Output of the Hidden layer before activation)

The GD for NN training is a.k.a. Back Propogation (BP)

SGD: Stochastic Gradient Descent(SGD)
In practice, the data are not given in same time, so cannot sum up from i to n in GD
Just change from i to n to i \belongsto I:

for $k = 1, 2, \cdots$

randomly choose $I \subset \{1, 2, \cdots, m\}$ with a small $|I|$.

$$\begin{cases} w^{(j,k+1)} = w^{(j,k)} - \alpha_k 2 c_j^{(k)} \sum_{i \in I} \left[\left(f_{w^{(k)},c^{(k)}}(x^{(i)}) - y_i \right) \sigma'\left(\langle w^{(j,k)}, x^{(i)} \rangle \right) \right] x^{(i)} \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad j = 1, \cdots, p. \\ c^{(k+1)} = c^{(k)} - \alpha_k \sum_{i \in I} 2 \left(f_{w^{(k)},c^{(k)}}(x^{(i)}) - y_i \right) \sigma\left((w^{(k)})^T x^{(i)} \right) \end{cases}$$

end

Deep Learning with deep neural networks

$$\min_{W,C} \sum_{i=1}^{N} \left(f_{W,C}(x^{(i)}) - y_i \right)^2 J W,$$