CH- optimization problem solution

→ Solving optimization problems : differentiation

We have seen optimization previously at 3 places.

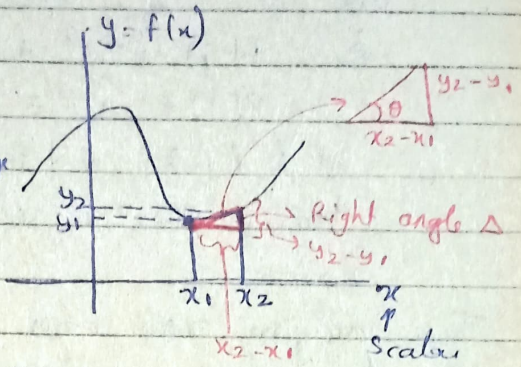
(i) PCA (ii) logistic Reg (iii) Linear Regression.

We use concepts like (i) Differentiation (ii) Maxima - minima

→ Single variable differentiation

so what does $\frac{dy}{dx}$ intuitively mean

$\frac{dy}{dx}$ means how much does y change as x change or we can say rate of change of y w.r.t x.



$\frac{y_2 - y_1}{x_2 - x_1}$ → change in y ⇒ $\frac{\Delta y}{\Delta x} = \tan\theta$

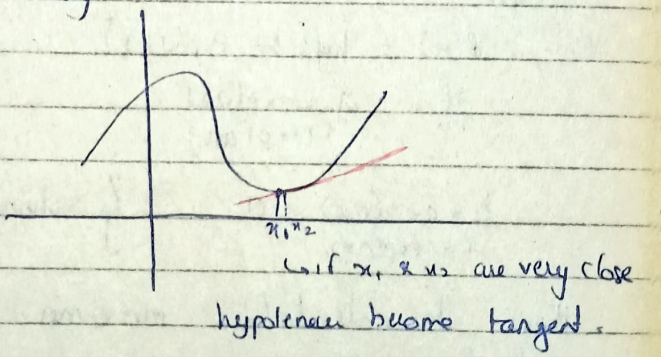
→ change in x

Now let's understand mathematically

$\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$

~~$\frac{dy}{dx}$ = tangent of~~ slope of.

$\frac{dy}{dx}$ = solope of tangent at f(x)



↳ if x_1 & x_2 are very close hypotenuse become tangent.

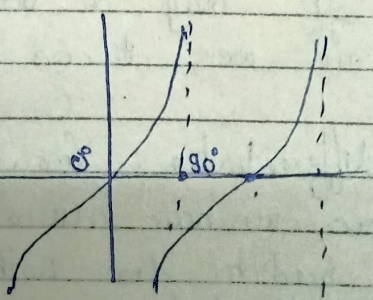
Let's understand Tan θ

$0 < \theta < 90° \Rightarrow$ +ve

Tan 0 = 0

Tan 90 = undefined

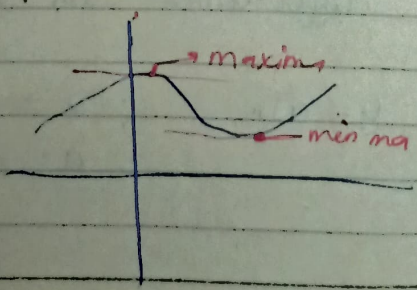
$90 < \theta < 180$ = -ve



* Maxima & Minima.

at both minma & maxima

slope = 0



Ex:- $f(x) = x^2 - 3x + 2$ find minima & maxima

$\frac{df}{dx} = 2x - 3$

$2x - 3 = 0$, $\boxed{x = \frac{3}{2}}$ → at $x = \frac{3}{2}$ slope = 0, so is this a maxima or minima.

One simple way is to compare x with 2 value one at $\frac{3}{2}$ and one at some other value.

Ex:- $f(\frac{3}{2}) = (\frac{3}{2})^2 + 3 \times \frac{3}{2} + 2$

$f(1) = 1 - 3 + 2$

If $f(\frac{3}{2}) < f(1)$ then $f(\frac{3}{2})$ is minima because there is some other value grater than $f(\frac{3}{2})$

There may be some case that func have no maxima or the func have no minima.

It is also possible that fun have many local max and min.

Ex:- $f(x) = \log(1 + \exp(ax))$

$\frac{df}{dx} = \frac{a * \exp(ax)}{1 + \exp(ax)}$

$\frac{a * \exp(ax)}{1 + \exp(ax)} = 0$ } solving this is not trivial.

Here to calculate maxima or minima just trying to find $\frac{df}{dx}$ may be very hard some time.

In such case we can use Gradient descent

→ Vector differentiation : Grad.

Till now we have assumed that x is scaler. But in ML we need to deal with vectors. So let's see how to perform differentiation on vectors.

- $y = a^T x$; $x = \langle x_1, x_2, \ldots, x_d \rangle$

$a = \langle a_1, a_2, \ldots a_d \rangle$ → constants.

$$\frac{df}{dx} = \nabla_x f$$

(grad or Del; vector)

$$\nabla_x f = \begin{bmatrix} \frac{df}{dx_1} \\ \frac{df}{dx_2} \\ \vdots \\ \frac{df}{dx_d} \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{bmatrix}$$

(Partial diff.)

$$y = a^T x = a_1 x_1 + a_2 x_2 + a_3 x_3 \dots + a_d x_d$$

$$\frac{\partial f}{\partial x_1} = a_1$$

$$\frac{\partial f}{\partial x_2} = a_2$$

$$\frac{\partial f}{\partial x_d} = a_d$$

$$\therefore \nabla_x (a^T x) = a \quad \text{(vector)}$$

$$L(w) = \sum_{i=1}^{n} \log\left(1 + \exp(-y_i w^T x_i)\right) + \lambda w^T w$$

$$\nabla_w L = \frac{(-y_i x_i)\exp(-y_i w^T x_i)}{1 + \exp(-y_i w^T x_i)} + 2\lambda w = 0$$

Now solving this is hard. we have to used gradient desc

Gradient Descent :- Now, let's learn gradient descent algori

It is an iterative algorithm.

x_0 ← first guess of x

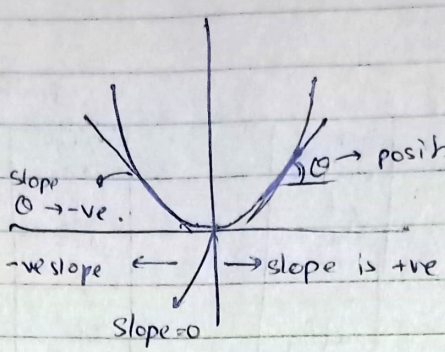
x_1 ← iteration 1

x_2 ← iteration 2

x_k ← k^{th} iteration

As our iteration number increases, we go closer to right x.

Slope changes its sign from +ve to -ve at minima.



Gradient descent Algo:-

① pick an initial pt = x_0 at random

② $x_1 = x_0 - \gamma \left[\frac{df}{dx}\right]_{x_0}$ → slope of the tangent:

constant: step size.

Let $\gamma = 1$ for now.

If $\frac{df}{dx} > 0$ then $\gamma * \frac{df}{dx} > 0$

$x_1 < x_0$

$\therefore x_0 - \gamma\left[\frac{df}{dx}\right] < x_0$

$\therefore$ we move closer.

If $\frac{df}{dx} < 0$ $\therefore$ $x_1 = x_0 + \gamma\left[\frac{df}{dx}\right]$

$x_1 > x_0$

again we move closer. because this time we are on other side

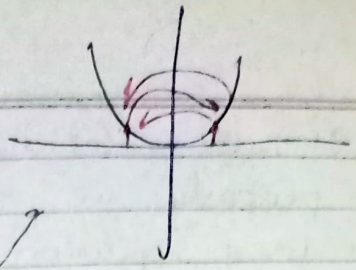
at i^{th} iteration

$$x_{i+1} = x_i + \gamma \left[\frac{df}{dx}\right]_{x_i}$$

at any iteration k

If $(x_{k+1} - x_k)$ is V. small then terminate at $x^* = x_k$

$$x_{i+1} = x_i - r * \left[\frac{\partial f}{\partial x}\right]_{x_i}$$



r is called learning rate or step size.

Some time we get oscillation problem due to ~~gradient~~ constant step size. To remove that we can change r at each iteration ~~with~~ any of the following techniq.

(i) reduce r with each iteration.

(ii) we can choose any func to change r.

* Gradient descent for Linear Regression.

$$w^* = \underset{w_0}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - w^T x_i)^2$$

↳ Not using w_0 & Regularization for simplicity.

$$L = \sum_{i=1}^{n} (y_i - w^T x_i)^2$$

of x_i, y_i are part of ~~training~~ Data hence they are ~~constant~~ vector of constant we do our optim. for w.

$$\nabla_w L = \sum_{i=1}^{n} \{2(y_i - w^T x_i)(-x_i)\}$$

(1) pick a random vector $w_0 = \langle \quad \rangle$

(2) $w_1 = w_0 - r * \sum_{i=1}^{n} (-2x_i)(y_i - w_0^T x_i)$

(3) $w_2 = w_1 - r * \sum_{i=1}^{n} (-2x_i)(y_i - w_1^T x_i)$

⋮

(k) $w_{k+1} = w_k - r * \sum_{i=1}^{n} (-2x_i)(y_i - w_k^T x_i)$

if $(w_{k+1} - w_k)$ ~~<0~~ return.

There is one problem here for every iteration we are computing the summation. If dataset is very large there is lot of time & computing power req.

for each iteration we are visiting whole data.

To solve this problem we use technique called SGD

✡ Stochastic Gradient descent (SGD)

previously we saw for linear regression

GD: $w_{j+1} = w_j - \gamma \times \sum_{i=1}^{n} (-2x_i)(y_i - w_j^T x_i)$

computing sum on whole data set at each iteration.

SDG:- $w_{j+1} = w_j - \gamma \times \sum_{i=1}^{(k)} (-2x_i)(y_i - w_j^T x_i)$

$1 \leq k \leq n$

pick random set of k-points from total dataset of n points and apply GD on these k-points will give same result.

It is called stocastic because we are using random val

if we pick smaller point i.e k is very less then no. of iteration increases.

k is also called batch-size.

and each iteration we pick differ random set of k points.

✡ Constrained optimization:-

Till now we have seen one optimization problems like $\max_x f(x)$ or $\min_x f(x)$ But PCA formulation is:

PCA: $\max_u \frac{1}{n} \sum_{i=1}^{n} (u^T x_i)^2$ s.t $u^T u = 1$

objective func^n — constraint.

These types of problem are called constrained optimization pro

General constraint func^n looks like:

$\max_x f(x)$ s.t $(g(x) = c)$ → equality constraint

$(h(w) \geq d)$ → inequality constraint.

So how to find maxima/minima of such type of funct.

$\max_x f(x)$ s.t $g(x)=c$, $h(x) \geq d$ → modify it using lagrangian multipliers

Lagrangian Multipliers.

$\mathcal{L}(x,\lambda,\mu) = f(x) - \lambda\{g(x)-c\} - \mu\{d-h(x)\}$

lagrangian multiplier

λ & μ are lagrangian multipliers. $\lambda \geq 0$ $\mu \geq 0$

$$\frac{\partial \mathcal{L}}{\partial x}=0 \; ; \; \frac{\partial \mathcal{L}}{\partial \lambda}=0 \; ; \; \frac{\partial \mathcal{L}}{\partial \mu}=0$$

↳ solve these equation: suppose we get $\tilde{x}$ after solving this then $\tilde{x}$ is the solution of the given optimization problem.

Now, let's revisit the PCA problem.

$$\text{PCA: } \max_u \frac{1}{n}\sum_{i=1}^{n}(u^T x_i)^2 \quad \text{s.t } u^T u = 1$$

$$\Downarrow$$

$$\max_u u^T S u \quad \text{s.t } u^T u = 1 \quad \& \quad S = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T$$

S is a constant (because it contains covariance) ← covariance matrix

$$\mathcal{L}(u,\lambda) = u^T S u - \lambda(u^T u - 1)$$

$$\frac{\partial \mathcal{L}}{\partial u} = 0 \Rightarrow \frac{\partial (u^T S u - \lambda u^T u - \lambda)}{\partial u} = 0$$

$$= Su - \lambda u = 0$$

$$Su = \lambda u \quad \text{// definition of eigen value \& eigen vector}$$

u is eigen vector of S

λ is eigen value of S

That is what we have done in PCA, we choosed the top eigen valued vectors of co-variance matrix. This is the proof.

Now, let's revisit Logistic Regression formulation in context of constrained optimization

LR:

$$w^* = \underset{w}{\text{argmin}} \left(\text{Logistic\_Loss} \right) + \lambda w^T w$$

Actual problem is, to minimize the logistic loss s.t w is normal to hyperplane and w is unit vector.

$$\therefore \quad w^* = \underset{w}{\text{argmin}} \left(\text{Logistic\_Loss} \right) \quad \text{s.t} \quad w^T w = 1$$

(Logistic_Loss: objective func; $w^T w = 1$: constraint)

Just use Lagrangian to solve this

$$\mathcal{L} := \text{logisticloss} - \lambda(1 - w^T w)$$

$$= \text{logisticloss} - \lambda + \lambda w^T w$$

$\therefore$ regularization can be thought of as imposing an constraint

★ Why does L_1 - regularization create sparsity.

while studying Logistic Reg:- we said that L_1 reg creates sparsity in w as compared to L_2-regularization. Now we will look at geometric intuition of it

L_2 regularization

$$\min_w \text{loss} + \lambda \|w\|_2^2$$

$$\lambda \|w\|_2^2$$

$$\min_w \|w\|_2^2$$

$$\min_w \left(w_1^2 + w_2^2 + \cdots + w_d^2 \right)$$

Let's choose only w_1 for now

$$\min_{w_1} w_1^2 = L_2(w_1)$$

L_1 regularization

$$\min_w \text{Loss} + \lambda \|w\|_1$$

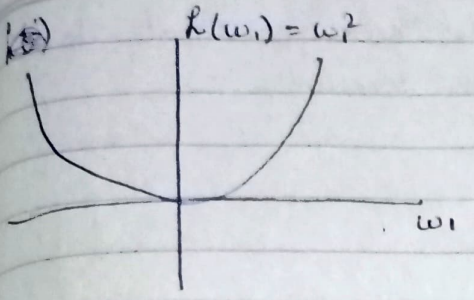
$$\lambda \|w\|_1$$

$$\min_w \|w\|_1$$

$$\min_w \left(|w_1| + |w_2| + \cdots + |w_d| \right)$$

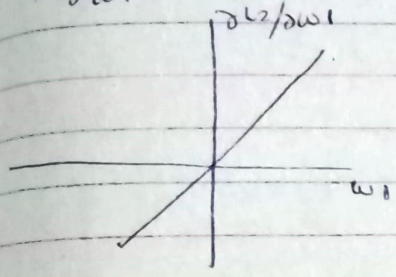
$$\min_{w_1} |w_1| = L_1(w_1)$$

L2



$L_2(w_1) = w_1^2$

$\frac{\partial L_2}{\partial w_1} = 2w_1$



Let $(w_1)_i$ is the value of w_1 in ith iteration of Gradient Descent

GD: $(w_1)_0, (w_1)_1 \dots (w_1)_K$

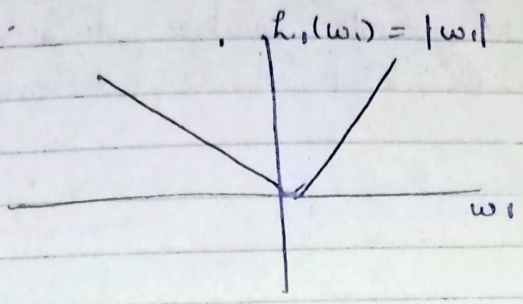
$$(w_1)_{i+1} = (w_1)_i - \gamma \left[\frac{\partial L_2}{\partial w_1}\right]_{w_1 i}$$

$(w_1)_{i+1} = w_1 i - \gamma * (2 * w_1 i)$

If $w_1 i$ is very small and γ is also very small

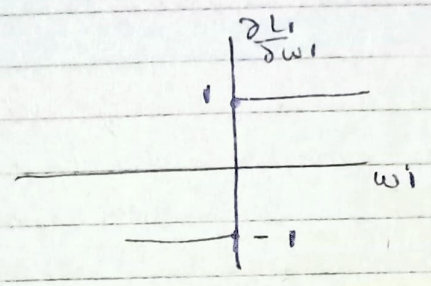
$\gamma * (2 * w_1 i) << \gamma$

L1



$L_1(w_1) = |w_1|$

$$\frac{\partial L_1}{\partial w_1} = \begin{cases} +1 & \text{if } w_1 > 0 \\ -1 & \text{if } w_1 < 0 \end{cases}$$



$$(w_1)_{i+1} = w_1 i - \gamma \left[\frac{\partial L_1}{\partial w_1}\right]_{w_1 i}$$

$(w_1)_{i+1} = w_1 i - \gamma [1]$ → assume $\frac{\partial L_1}{\partial w_1} = +1$

Hence in each iteration change in $(w_1)_{i+1}$ in L_2 is very less as compared to L_1 where change is more. Hence Rate of convergence is different in both.

All is this happen due to $\frac{\partial L_1}{\partial w_1}$ is a constant i.e. +1 or -1 whereas as $\frac{\partial L_2}{\partial w_2}$ is not constant but reduces with increase in no. of iteration Hence L_2 converges very slow.

L_2 regularization change the value of w_1 from one iteration to another

L_1 regularization constantly reduce w_1 towards w_1^*

∴ L_1 create sparsity as because L_1 has constant slope while slope of L_2 decreases with iteration.