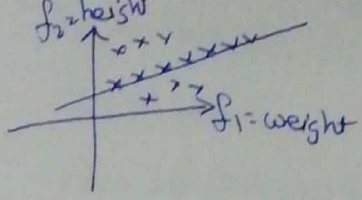
25.1 Geometric intuition of Linear Regression

Eg: Predict height given weight, gender, ethnicity, haircolor.

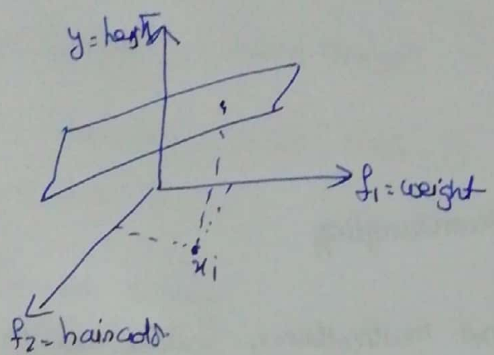
Lin Reg: find a line that fits the given data



$y = w_1 * \text{weight} + w_0$

$y = mx + c$

3 dimensional space:

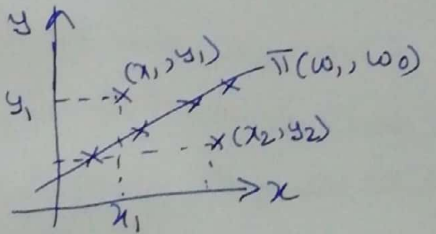


$\text{height} = w_1 \times f_1 + w_2 \times f_2 + w_0$

$y_i = w_1 x_i + w_2 x_{i2} + w_0$

$$\boxed{y = w^T x_i + w_0}$$

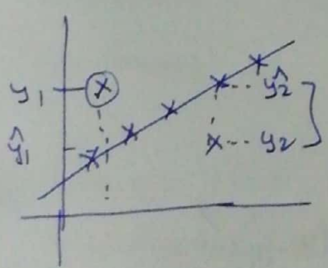
find a line/plane that best fits the datapoints



Error for x_1 = $y_1 - \hat{y}_1$

Error for x_3 = $y_2 - \hat{y}_2$

25.2 Mathematical Formulation



$\left.\begin{array}{l} \text{Error1} = y_1 - \hat{y}_1 = +ve \\ \text{Error2} = y_2 - \hat{y}_2 = -ve \\ \text{Error3} = 0 \end{array}\right\}$ we have to take a square

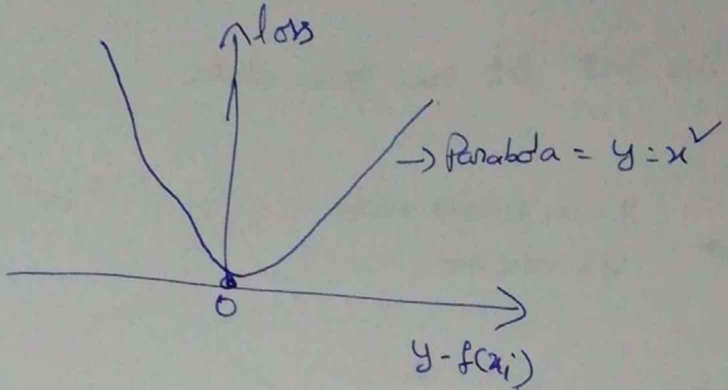
we have to find the optimal $\underset{\text{vector}}{\vec{w}^*}$, $\underset{\text{scalar}}{w_0^*}$ =

$$\underset{w, w_0}{\text{argmin}} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \qquad \hat{y}_i = f(x_i) = w^T x_i + x_0$$

$$(w^*, w_0^*) = \underset{w, w_0}{\text{argmin}} \sum_{i=1}^{n} \{y_i - (w^T x_i + x_0)^2\}$$

Regularization

$$(w^*, w_0^*) = \underset{w, w_0}{\text{argmin}} \sum_{i=1}^{n} \overbrace{\{y_i - (w^T x_i + w_0)\}^2}^{\text{square loss}} + \underbrace{\lambda \|w\|_2^2}_{L_2\text{-reg}}$$



25.3 Real world cases

Imbalanced data : upsampling or downsampling

feature importance & Interpretability : features are not multicollinear, feature weights can be used.

feature Engineering & feature transformation :

Reg : L_1 reg , $\lambda_1 \uparrow$

Outliers : Log-reg : sigmoid function → limiting the impact of outliers

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

sq loss is highly impacted by outliers

$D_{train} \longrightarrow w^*, w_0^*$

RANSAC (Random sampling)

- ↳ find pts very far away for π $(y_i - \hat{y}_i) \uparrow$
- ↳ remove these pts as outliers
- ↳ $D_{train} = D_{train} -$ outlier
- ↳ repeat if needed

25.4 Code sample for Linear Regression.

```
from sklearn.datasets import load_boston
boston = load_boston()

// (506, 13)

boston. ~~datasize~~ DESCR

import pandas as pd

bos = pd.DataFrame (boston.data)

print (bos.head ())

bos ['Price'] = boston.target

X = bos.drop ('PRICE', axis = 1)
Y = bos ['price']

import sklear.

// train & split use the code

from sklearn.linear_model import LinearRegression.
lm = LinearRegression ()
lm.fit (X_train, Y_train)
Y_pred = lm.predict (X_test)

plt.scatter (Y_test, Y_pred)
plt.xlabel ("Prices: $Y_i$")
plt.ylabel ("Predicted Prices")
plt.show()

delta_y = Ytest _ Y_pred
import seaborn as sns;
import numpy as np
sns.set_style ('white grid')
sns.kdeplot (np.array (delta_Y), bw=0.5)
plt.show ()
```

26.1 Differentiation

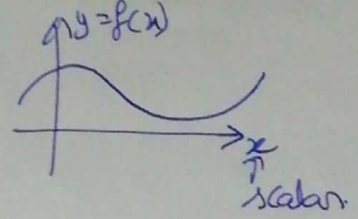
Solving optimization problems : differentiation.

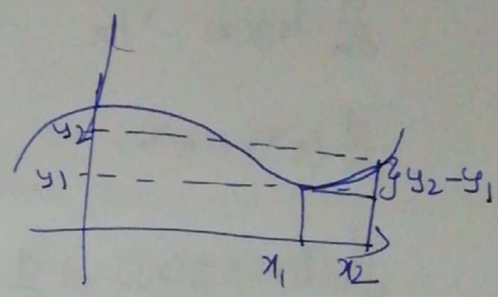
ML → differentiation → Single value, vector

↳ maxima & minima.

Single variable diff :-

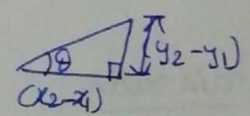
$y = f(x)$





$\left\{ \frac{dy}{dx} = \frac{df}{dx} = y' = f' \right\}$

↳ differentiation of y w.r.t x.



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

θ : Slope

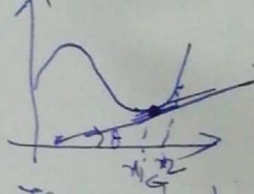
$\frac{dy}{dx}$ → Intuitively

↳ Rate of change of y as x changes.

↳ how much does y changes as x changes.

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

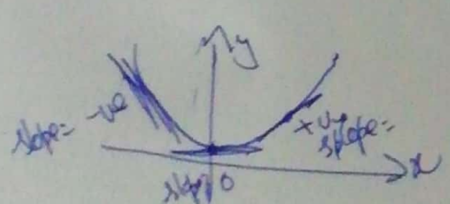
when x_2 comes close to x_1 then Δx will be almost close to zero



Tangent is hypotenuse that we obtain as $\Delta x \to 0$

$\tan\theta = \frac{dy}{dx}$

$\left[\frac{dy}{dx}\right]_{x_1}$: Slope of the tangent to $f(x)$ @ $x = x_1$



basic

$$\frac{d}{dx}(x^n) = nx^{n-1} \qquad \frac{d}{dx}(x^2) = 2x$$

$$\frac{d}{dx}(c) = 0 \qquad \frac{d}{dx}(3x) = 3$$

Constant

$$\frac{d}{dx}(cx^n) = cnx^{n-1}$$

$$\frac{d}{dx}\log(x) = 1/x$$

$$\frac{d}{dx}(e^x) = e^x$$

$$\frac{d}{dx}(f(x)+g(x)) = \frac{d}{dx}f(x) + \frac{d}{dx}g(x)$$

chain rule

$$\frac{d}{dx}f(g(x)) = \frac{df}{dg}\cdot\frac{dg}{dx}$$

Ex $f(g(x)) = (a-bx)^2$

$g(x) = a-bx$

$f(x) = x^2$

$$\frac{d}{dx}f(g(x)) = \frac{df}{dg}\cdot\frac{dg}{dx}$$

$$\frac{dg}{dx} = \frac{d}{dx}(a-bx) =$$

$$= \frac{d}{dx}(a) - \left(x\frac{d}{dx}(b) + b\frac{d}{dx}x\right)$$

$$= 0 \qquad 0 \qquad 1$$

$$= -b$$

$\frac{df}{dg}$: $g(x) = z$

$f(g(x)) = z^2$

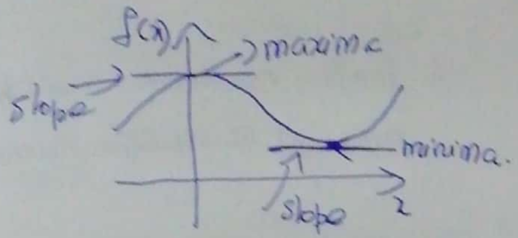
$$\frac{df}{dg} = \frac{dz^2}{dz} = 2z$$

$$\frac{d}{dx}(f(g(x))) = -2b(a-bx)$$

26.3 Online differentiation tools

https://www.derivative-calculator.net.

26.4 Maxima & minima



At minima and maxima slope will become zero

Slope is nothing but tan of angle b/w tangent & x axis.

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

$\frac{df}{dx} = 0 \Rightarrow$ slope $= 0$

$\frac{df}{dx} = 2x - 3 = 0 \Rightarrow \boxed{x = 3/2}$ At $x = 3/2$ slope $= 0$

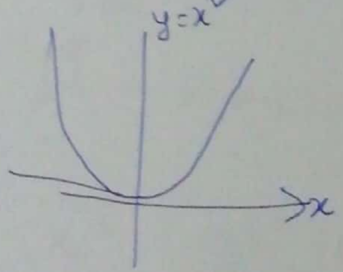
(Q) Is this a maxima or minima.

$f(1.5) = (1.5)^2 - 3(1.5) + 2 = -0.25$

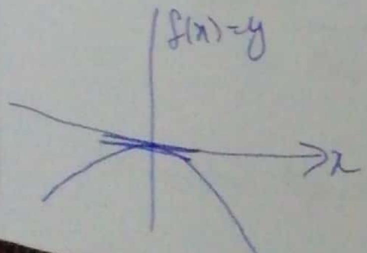
$f(1) = 0$ ← take a value close to x.

$f(1.5) < f(1)$

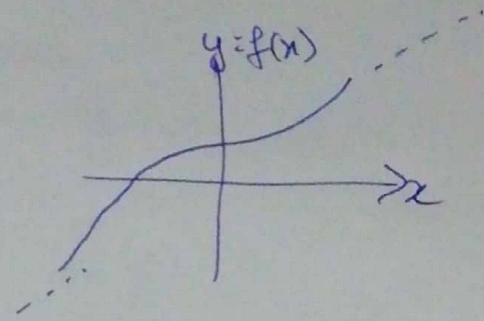
$f(1.5)$ cannot be maxima, it has to be minima



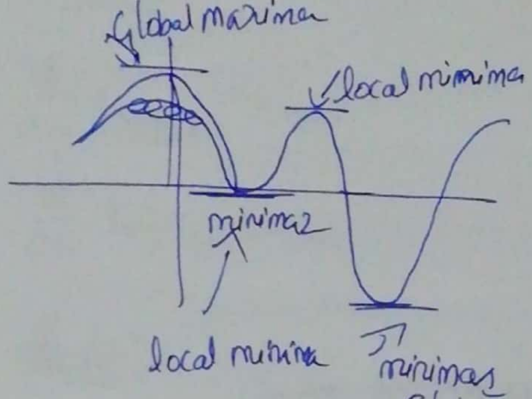
→ no maxima, it has minima at $x=0$



→ It has maxima but no minima



There is no maxima & no minima because as the dotted lines imply they can go to maxima & minima.



A function can have multiple minimal & multiple maxima.

minima of all the minima is called Global minima

<u>Imagine</u>

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

find minima & maxima.

$$\frac{df}{dx} \quad \frac{a \cdot \exp(ax)}{1 + \exp(ax)} = 0$$

Solving this is not trivial & hard. To solve this we have Computerised program called $\boxed{\text{Gradient descent}}$ to solve maxima & minima

26.5 Vector Calculus : Grad.

$$y = a^T x$$

$x = \langle x_1, x_2 \cdots x_d \rangle$

$a = \langle a_1, a_2 \cdots a_d \rangle$ Constant

$$f(x) = y = \sum_{i=1}^{d} a_i x_i$$

→ grad or del

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

vector

$$\nabla_x f = \begin{bmatrix} \frac{df}{dx_1} \\ \frac{df}{x_2} \\ \vdots \\ \frac{df}{dx_d} \end{bmatrix} \in \mathbb{R}^d$$

→ resultant also is vector

$$\frac{df}{dx_1} = \frac{\partial f}{\partial x_1}$$ → Partial differentiation

$$f(x) = y = a^T x = \sum_{i=1}^{d} a_i x_i$$

$$\nabla_x f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{bmatrix} : \frac{\partial f}{\partial x_1} = a_1, \quad \frac{\partial f}{\partial x_d} = a_d = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_d \end{bmatrix}$$

$$\nabla_x (a^T x) = a$$

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

logistic loss

x_i & y_i are constants ← Data

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

Solving this Equation is not very Easy

26.6 Gradient descent: geometric intuition

when dealing with maxima & minima

$\frac{df}{dx} = 0$; $\nabla_x f = 0$

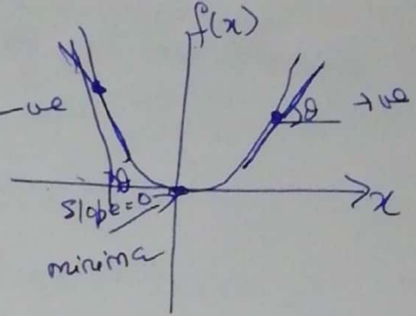
→ Gradient Descent

↳ Iterative algorithm.

↳ pick a random number x_0 ← first guess of x^*

↳ using gradient descent we will move to another point x_1

Eventually we will move closer to x^*



$x^* = \underset{x}{\arg\min}\ f(x)$

$\min f(x) = \max -f(x)$

(or)

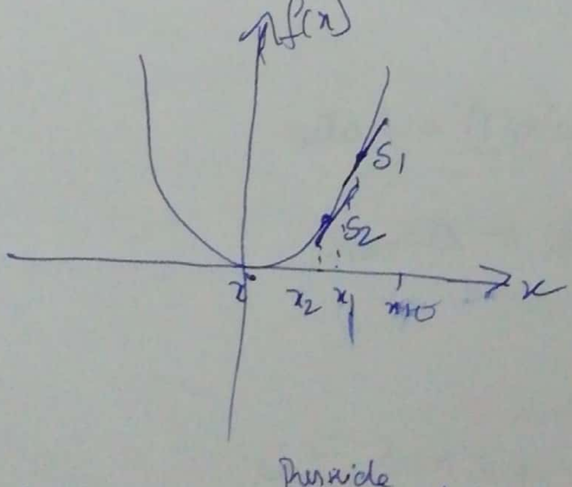
$\max f(x) = \min -f(x)$

minima

one side:- slope = +ve

other side :- slope = -ve.

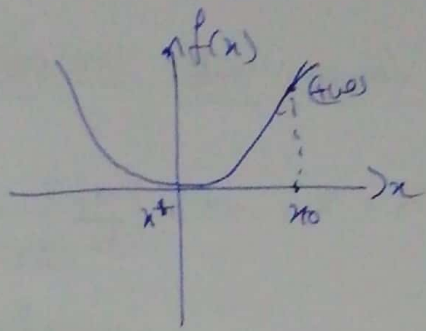
→ slope changes its sign fr +ve to -ve @ minima



Approaching minima ←(this side) slope decreases

Approaching minima → slope increases

Gradient Descent



① Pick an initial pt = x_0 @ random

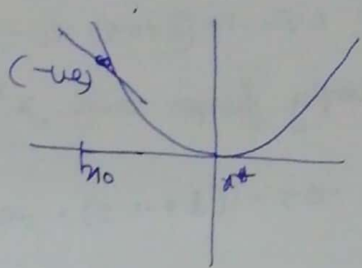
② we have to pick x_1 such that x_1 is closer to x^* than x_0

$$x_1 = x_0 - \underbrace{\gamma}_{\text{stepsize}} \left[\frac{df}{dx}\right]_{x_0}$$

Let step-size = 1

$$x_1 = x_0 - 1(+ve) \Rightarrow x_1 < x_0$$

$$x_1 = x_0 - \underbrace{\gamma}_{(=1)} \underbrace{\left[\frac{df}{dx}\right]_{x_0}}_{-ve}$$



$$x_1 = x_0 - 1(-ve)$$

$$\boxed{x_1 > x_0}$$

③ $x_2 = x_1 - \gamma \left[\frac{df}{dx}\right]_{x_1}$

At Any iteration

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

$x_0, x_1, x_2 \ldots x_k$

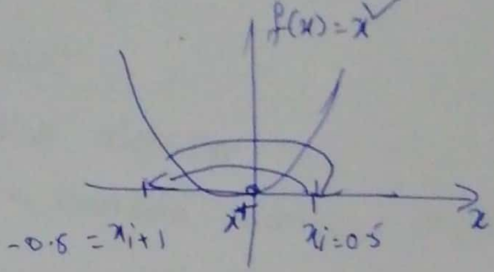
$$\boxed{x_k = x_{k-1} - \gamma \left[\frac{df}{dx}\right]_{x_{k-1}}}$$ ← update function.

If $(x_{k+1} - x_k)$ is very small

Then terminate @ $x^* = x_k$

26.7 Learning rate (or) step size

$$x_i = x_{i-1} - r * \left[\frac{df}{dx}\right]_{x_{i-1}}$$



$$x_{i+1} = x_i - r * \left[\frac{df}{dx}\right]_{x_i} \qquad (r=1)$$

$$x_{i+1} = 0.5 - 1*(2*0.5) = -0.5$$

we simply jump over x^*

$$x_{i+2} = -0.5 - 1(2*-0.5) = -0.5 - 1(-1) = 0.5$$

remedy for oscillation:

→ change r with each iteration

→ reduce r for each iteration

s.t iterations ↑ and r ↓

26.8 Gradient descent for linear regression

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \underbrace{w^T x_i}_{}) ^2 \underbrace{\quad}_{\text{no-reg}}$$

Not using w_0 (intercept)

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

$\because$ x_i & y_i are Constants

Vector Scalar

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

Pick a random vector $w_0 = \langle \cdots \rangle$

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

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

we will calculate $w_0, w_1, w_2 \cdots w_k \; w_{k+1}$

when w_{k+1} & w_k is very small then we will declare $w^* = w_k$

Problem

In n is very large $n = 1$ million, computing summation is very Expensive.. Gradient descent will become very slow

Solution

Stochastic Gradient descent is the solution.

26.9 SGD algorithm

Linear regression :-

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

SGD (Stochastic Gradient Descent)

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

$$1 \le k \le n$$

→ Pick a random set of k-pts

$$w^*_{GD} = w^*_{SGD}$$

(n=1 million)
GD:- 100 iterations to Converge at x^*

SGD:- n=1000 , in >100 iterations x^*

SGD:- n=10 ; in >1000 iteration to Converge at x^*

K = # of points pick of iteration.
Every iterations points in K is different

K = $\underbrace{\text{batch size}}$ in SGD
batch of random pts

SGD:- The most imp-optimization algo in ML

26.10 Constrained Optimization & PCA

Till now we have seen optimization problems like

$\max_x f(x)$ & $\min_x f(x)$

If we recall PCA

$$\max_u \underbrace{\frac{1}{n}\sum_{i=1}^{n} (u^T x_i)^2}_{\text{objective}}$$

$$\text{s.t } \underbrace{u^T u = 1}_{\text{Constraint}} \qquad \approx \qquad \max_x f(x) \quad \text{s.t } g(x) = c$$

general Constraint optimization.

$$\max_x \underbrace{f(x)}_{\text{objective function}}$$

s.t $g(x) = c$ → Equality Constraint

$h(x) \geq d$ → inEquality Constraint

$k(x) \leq e \rightarrow -k(x) \geq -e$

can be converts by multiplying both side with negative sign.

Lagrangian multipliers

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

λ & u ~~is~~ are called Lagrangian multipliers.

$\lambda \geq 0$, $u \geq 0$

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

$$\downarrow \qquad \downarrow \qquad \downarrow$$

$$\tilde{x} \qquad \tilde{\lambda} \qquad \tilde{u}$$

$x^* = \max_x f(x)$

Here

$\tilde{x} = x^*$

PCA :-

$$\left.\begin{array}{l} \max_{u} \frac{1}{n} \sum_{i=1}^{n} (u^T x_i)^2 \\ \text{s.t } u^T u = 1 \end{array}\right\} \rightarrow \begin{array}{l} \max_{u} \; u^T S u \\ \text{s.t } u^T u = 1 \end{array}$$

&

$$S = \frac{1}{n} \sum_{i=1}^{n} x_i x^T$$

covariance matrix of X

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

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

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

$$= Su = \lambda u$$

covariance matrix

Vector.

26.11 Logistic Regression formulation revisited

$$w^* = \underset{w}{\text{argmin}}\ (\text{logistic-loss}) + \text{regularization}\ (\lambda w^T w)$$

Lagrangian

$$\mathcal{L} = \text{logistic loss} - \lambda(1 - w^T w)$$
$$= \text{logistic loss} - \lambda + \lambda w^T w$$

regularization can be thought as imposing an Eq. Constraint.

26.12 Why L1 regularization creates sparsity?

Logistic regression: L1 reg creates sparsity in w as compared to L2 regularization.

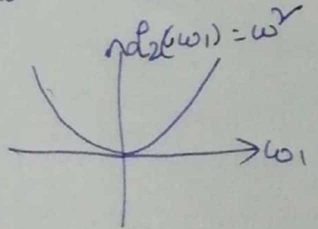
Sparsity means: $W \in \langle w_1, w_2, \dots w_d \rangle$
most of w_i's are "0"

$\mathcal{L}_2$

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

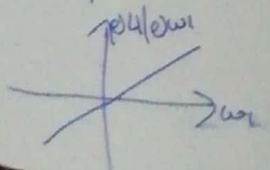
$\min_{w_1, w_2 \dots w_d} (w_1^2 + w_2^2 + \dots + w_d^2)$

$\min_w w_1^2 = \mathcal{L}_2(w_1)$



$\mathcal{L}_2(w_1) = w_1^2$

$\frac{\partial \mathcal{L}_2}{\partial w_1} = 2w_1$

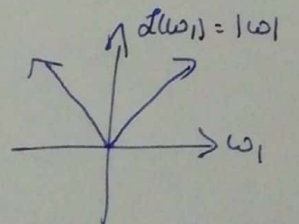


$\mathcal{L}_1$

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

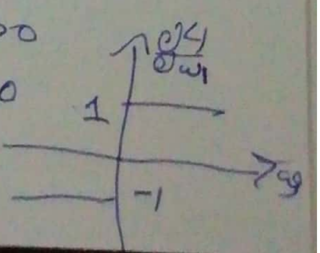
$\min_{w_1, w_2 \dots w_d} (|w_1| + |w_2| + \dots |w_d|)$

$\min_w |w_1| = \mathcal{L}_1(w_1)$



$$\mathcal{L}_1(w_1) = |w_1| = \begin{cases} w_1 & \text{if } w_1 > 0 \\ -w_1 & \text{if } w_1 \le 0 \end{cases}$$

$$\frac{\partial \mathcal{L}_1}{\partial w_1} = \begin{cases} +1 & \text{if } w_1 > 0 \\ -1 & \text{if } w_1 < 0 \end{cases}$$



L_2

$(w_1)_{j+1} = (w_1)_j - \eta\left[\frac{\partial L_2}{\partial w_1}\right]_{w_{1j}}$

Let $(w_1)_j$ = +ve

$(w_1)_{j+1} = (w_1)_j - r(2w_{1j})$

L_1

$(w_1)_{j+1} = (w_1)_j - \eta\left(\frac{\partial L_1}{\partial w_1}\right)_{w_{1j}}$

Let $(w_1)_j$ = +ve.

$(w_1)_{j+1} = (w_{1j}) - r(1)$

26.13 Implement SGD for linear regression