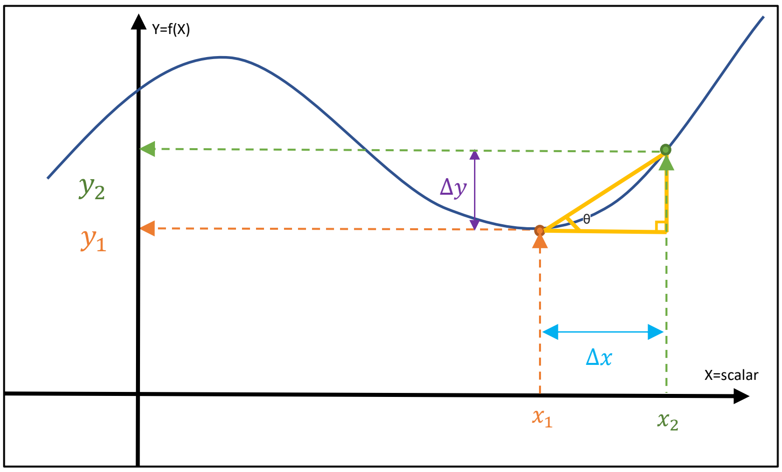
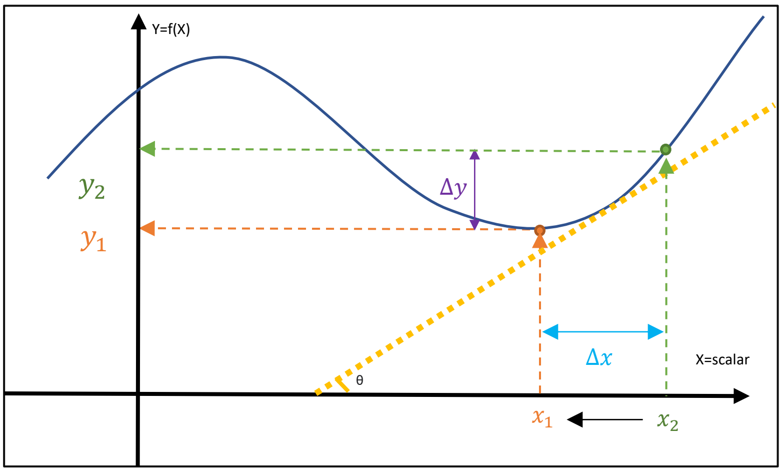
- **Differentiation**
  - Single variable differentiation

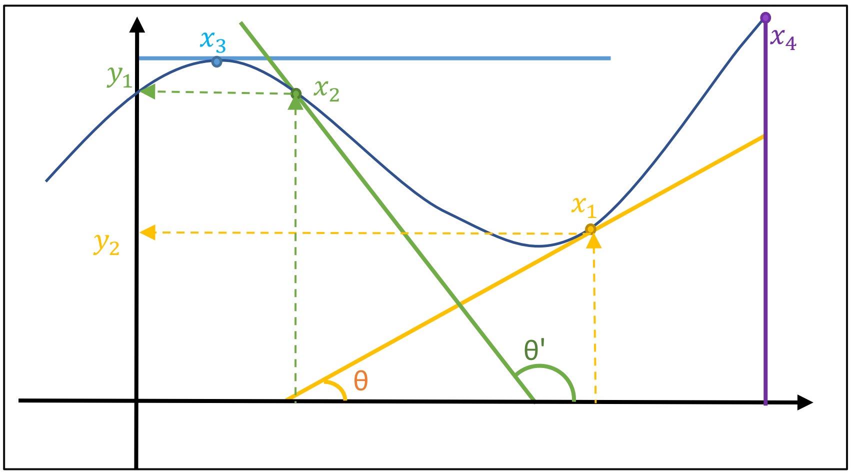


  - Assuming x_1 & x_2 are very close, Differentiation is rate of change of y w.r.t x in the vicinity of x_1.
  - Geometrically, $\boxed{\frac{y_2 - y_1}{x_2 - x_1} = \frac{\Delta y}{\Delta x} = tan\theta}$
  - Mathematically, $\frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$
  - Derivative between y & x is rate of change of y w.r.t. x as $\Delta x \to 0$. As x_2 comes closer towards x_1 then only $\Delta x \to 0$
  - As $\Delta x \to 0$ i.e., difference between x_1 & x_2 tends to be 0. Since the triangle will be so small here the hypotenuses become tangent touching x_1 . we calculate tan θ to get derivative where θ is angle between tangent & x axis



$$\boxed{\begin{array}{l} \frac{dy}{dx} = \lim_{\Delta x \to 0} \frac{\Delta y}{\Delta x} = tan\theta = slope\ of\ tangent\ to\ f(x) \\ \quad \frac{dy}{dx_{x1}} = slope\ of\ tangent\ to\ f(x) given\ x = x1 \end{array}}$$

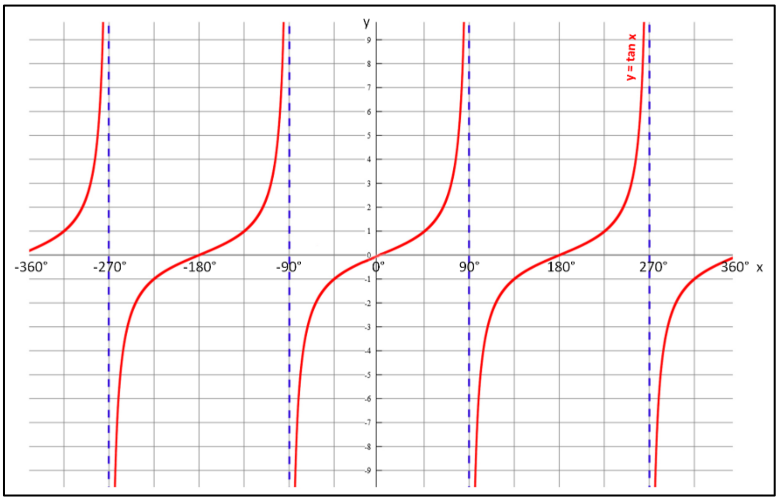
  - Similar to x_1 we can have x_2, x_3, x_4. We calculate tan θ of tangent to curve with x axis at these points.



- ○ Basic Mathematical functions and their derivatives

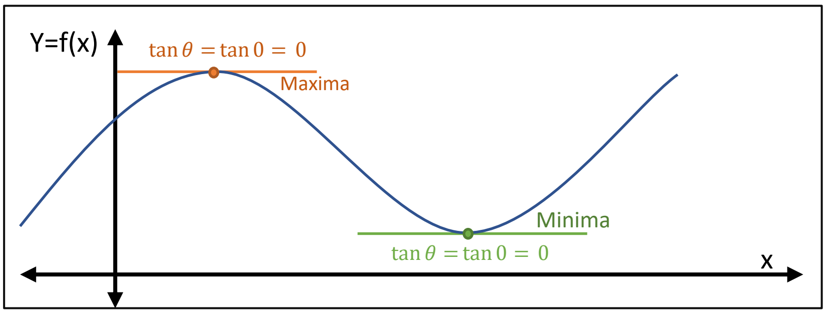
function	derivative
x^n	nx^{n-1}
$constant$	0
cx^n	cnx^{n-1}
$\log x$	$1/x$
e^x	e^x
$f(x) + g(x)$	$\frac{d}{dx}f(x) + \frac{d}{dx}g(x)$
$f\{g(x)\}$	$\frac{df}{dg}\cdot\frac{dg}{dx}$

- ○ Interpretation of $tan\theta$

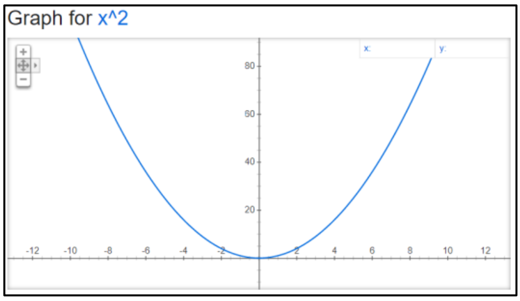


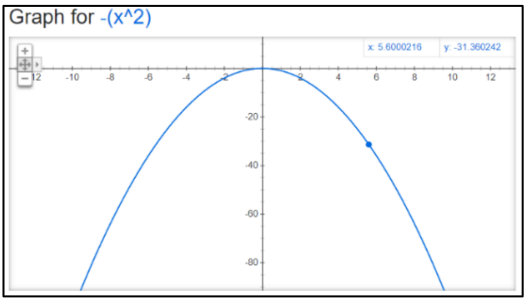
θ	$\tan\theta$
0° - 90°	>0
0	0
90°	∞
90° - 180°	<0

- **Maxima and Minima**

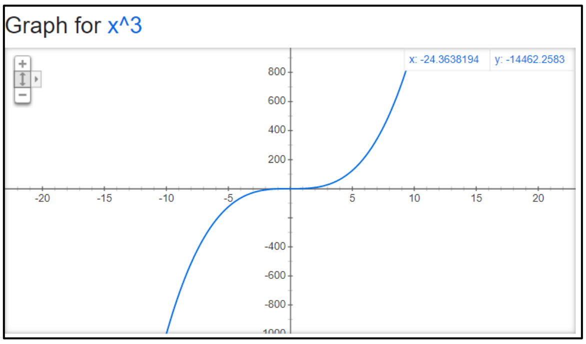


  - ○ The maxima and minima of a function, known as extrema, are the largest and smallest value of the function.
    - ▪ $f(x) = x^2 - 3x + 2$
    - ▪ $\frac{df}{dx} = 2x - 3$
    - ▪ $x = \frac{3}{2}$ at
    - ▪ $f(x = 1.5) = 1.5^2 - 3 * 1.5 + 2 = -0.25$
    - ▪ $f(x = 1) = 1^2 - 3 * 1 + 2 = 0$
  - ○ So, we have found out that at x = 1 slope of tangent is 0. Now 1.5 can be either maxima and minima
  - ○ But $f(x = 1) > f(x = 1.5)$So, 1.5 is minima
- **Special cases**
  - ○ For every function, it's not necessary that it has maxima and minima.
  - ○ $f(x) = x^2$ doesn't have maxima and $f(x) = -x^2$doesn't have minima

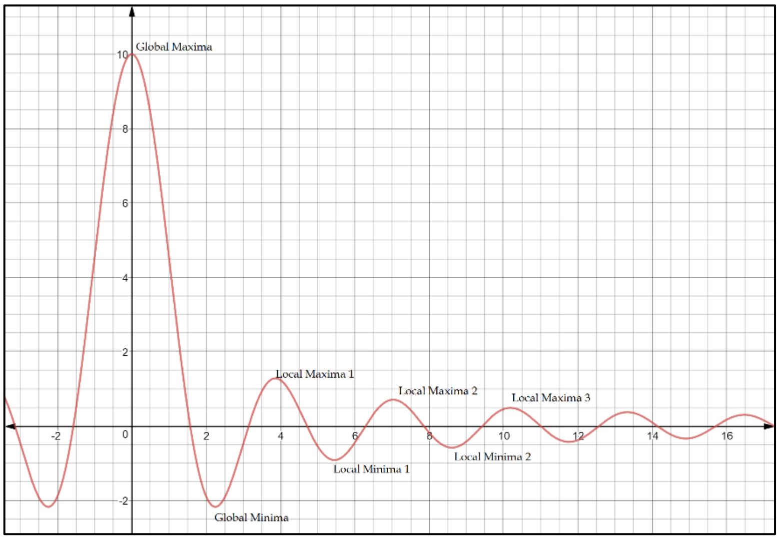

Graph for x^2


Graph for -(x^2)

- No Maxima or Minima



- Multiple Minima & Maxima



- **Vertor calculus: Grad or Dell** (∇)
  - If x is high dimensional vector

$$\boxed{f(x) = y = a^T x = \sum_{i=1}^{d} a_i x_i \mid a \; is \; constant}$$

  - The derivative of f(x) where x is a vector,

$$\boxed{\frac{df}{dx} = \nabla_x f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{bmatrix}}$$

$$\boxed{\sum_{i=1}^{n} a_i x_i = a_1 x_1 + \cdots + a_d x_d}$$

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

$$\boxed{\nabla_x f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_d} \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_d \end{bmatrix} = a}$$

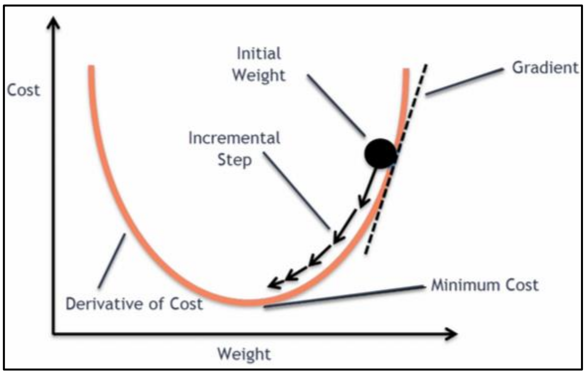
  - If x as scalar,

$$\boxed{\frac{d}{dx}(ax) = a}$$

- **Gradient descent**
  - To solve an optimization problem $f(x)$, we want to find optimum value x where minima will be 0. But such equations are non-trivial to solve.

$$\boxed{\frac{df}{dx} = 0 \ or \ \nabla_x f = 0}$$

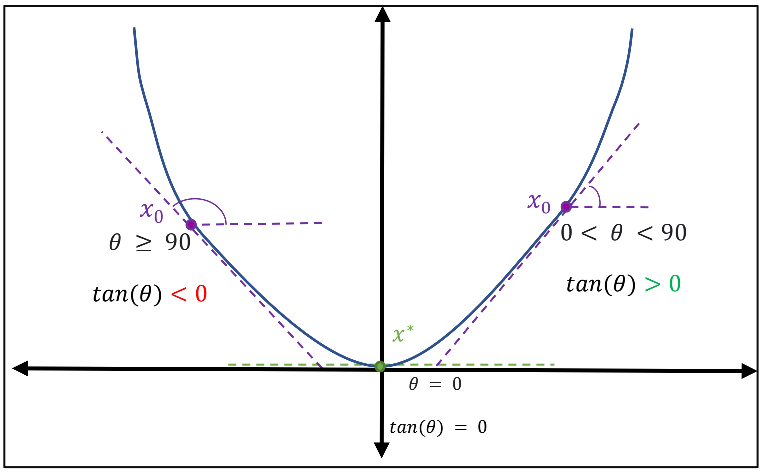
  - Gradient descent is a first-order iterative optimization algorithm for finding a local minima x^* of a differentiable function.



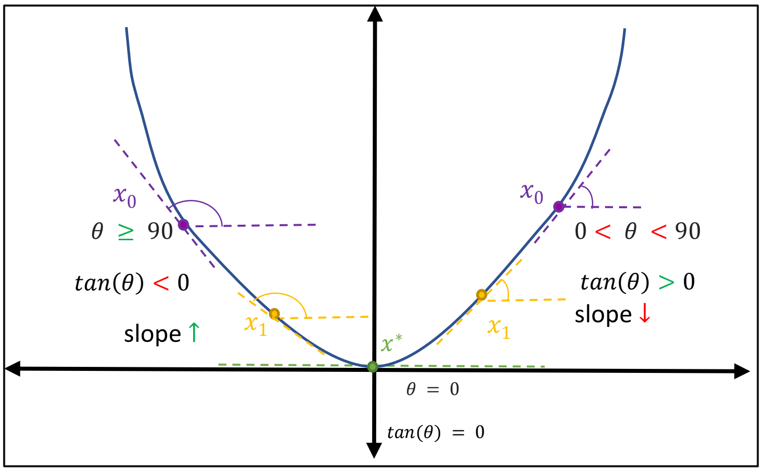
  - Our main goal is to find $\boxed{x^* = argmin_x\ f(x)}$. It's easy to fund maxima if we know minima

$$\boxed{\begin{array}{l} minima(f(x)) = maxima(-f(x)) \\ maxima(f(x)) = minima(-f(x)) \end{array}}$$

  - We pick x^* randomly and make it x_0
    - after 1st iteration, we get a value x_1
    - after 2nd iteration, it will be x_2
    - Finally, we reach x_k which is very close to x^*
  - After each iteration we get closer to x^*

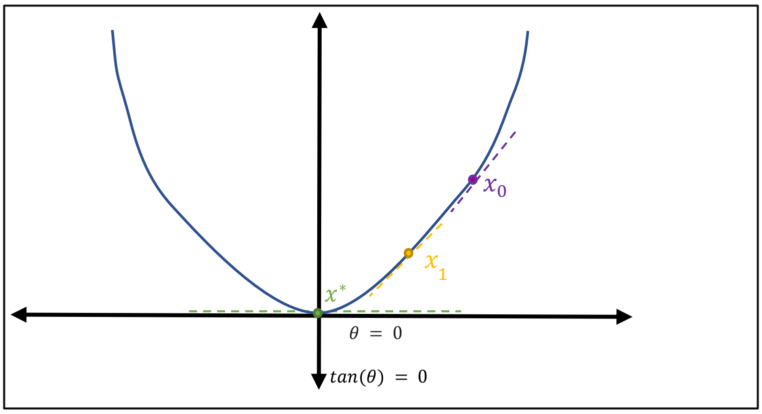


  - On the left side of minima, tan $\theta > 0$ & on right of its tan $\theta < 0$. At minima point, slope changes its sign



  - On the left side of minima, as the point starts reaching minima, slope ↑
  - On the right side of minima, as the points starts reaching minima slope ↓
  - In Gradient descent, we pick a first point randomly x_0

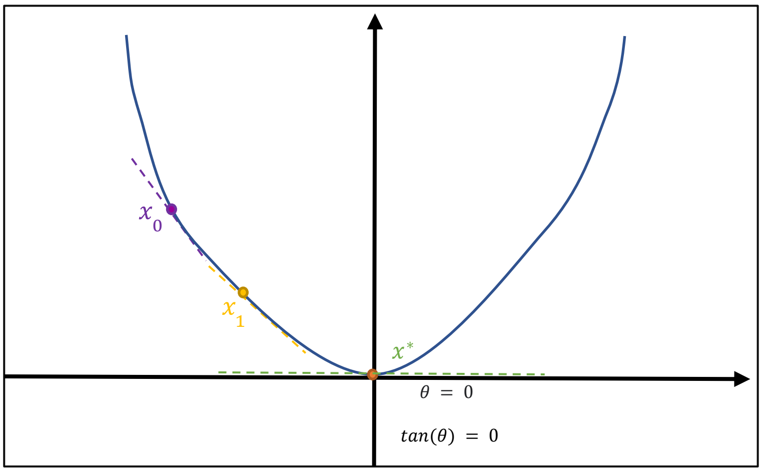
- **Case1**- x_0 is on right side of minima.



- Now x_1 has to be calculated such that it will be closer to x^*

$$\boxed{x_1 = x_0 - \eta \left[\frac{df}{dx}\right]_{x_0} \mid \eta = stepsize\ (> 0)}$$

- Let's say $\eta = 1$ and $\boxed{\left[\frac{df}{dx}\right]_{x_0} = \tan\theta > 0}$
- As $\eta \left[\frac{df}{dx}\right]_{x_0} > 0$, $x_1 < x_0$
- E.g., say $x_0 = 5$ & $\left[\frac{df}{dx}\right]_{x_0} = 0.5$, $x_1 = 5 - 1 * 0.5 = 4.5$
- **Case2**- x_0 on left side of minima



- Let's say $\eta = 1$ and $\boxed{\left[\frac{df}{dx}\right]_{x_0} = \tan\theta < 0}$
- As $\eta \left[\frac{df}{dx}\right]_{x_0} < 0$, $x_1 > x_0$
- E.g., say $x_0 = -5$ & $\left[\frac{df}{dx}\right]_{x_0} = 0.5$, $x_1 = -5 - 1 * -0.5 = -4.5$
- In the same iterative way, we keep on finding $x_2, x_3, \ldots \ldots, x_k$

$$\boxed{x_k = x_{k-1} - \eta \left[\frac{df}{dx}\right]_{x_{k-1}}}$$

- If $x_{k+1} - x_k$ is very small, then terminate and we conclude,

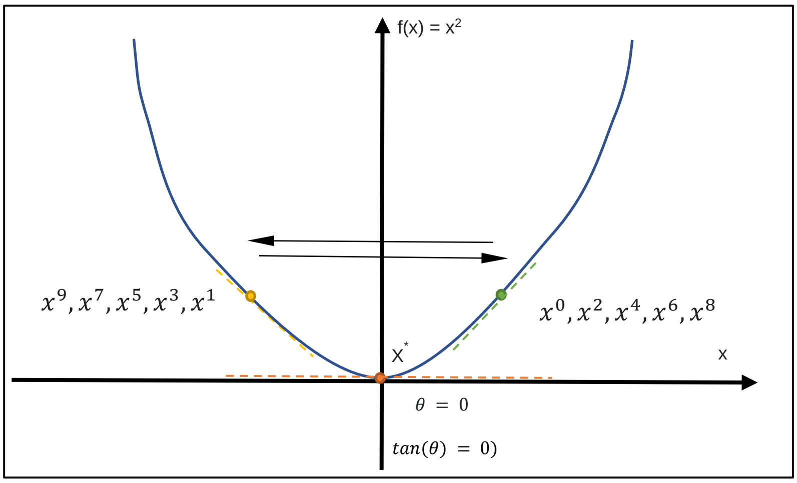
$$\boxed{x_* = x_k}$$

- To conclude the observations,

$$\boxed{\left[\frac{df}{dx}\right]_{x_0} \geq \left[\frac{df}{dx}\right]_{x_1} \geq \left[\frac{df}{dx}\right]_{x_2} \geq \ldots \ldots \ldots \ldots \left[\frac{df}{dx}\right]_{x_{k-1}} \geq \left[\frac{df}{dx}\right]_{x_k}}$$

- The jump between two points will be reduced as we move ahead

- **Learning rate or step-size**



  - Let's say $x_i = 0.5, f(x) = x^2, \eta = 1$
  - $x_{i+1} = x_i - \eta \left[\frac{df}{dx}\right]_{x_i} = 0.5 - 1 * (2 * 0.5) = -0.5$
  - $x_{i+2} = x_{i+1} - \eta \left[\frac{df}{dx}\right]_{x_{i+1}} = -0.5 - 1 * (2 * -0.5) = 0.5$
  - So, we are not converging towards x^* as it's each x are getting oscillated between 0.5 & -0.5.
  - The solution to this problem is we change the learning rate or step size at each of iteration
  - We need to find a function $h(i)$ such that

$$\boxed{\eta = h(i)\ |i = iteration\ number\ s.t\ as\ i \uparrow \eta \downarrow}$$

- **Gradient descent for linear regression**

$$\boxed{(W^*, W_0^*) = argmin_{w,w0} \sum_{i=1}^{n} \{y_i - (W^T x_i)\}^2}$$

  - Here, we are trying to minimize w & we will derive the formula first without considering regularization term x_0
  - Here x_i & y_i are constants and we need to find minimum W our function $L(W)$ for linear regression

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

  - Since, W^T is a vector, we need to calculate vector calculus

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

  - By applying Gradient Descent, we will pick random W_0

$$W_1 = W_0 - \eta * \textstyle\sum_{i=1}^{n}\{-2x_i\}\{y_i - (W_0^T x_i)\}$$
$$W_1 = W_1 - \eta * \textstyle\sum_{i=1}^{n}\{-2x_i\}\{y_i - (W_1^T x_i)\}$$
$$\vdots$$
$$W_k = W_{k-1} - \eta * \textstyle\sum_{i=1}^{n}\{-2x_i\}\{y_i - (W_{k-1}^T x_i)\}$$

  - We will compute this till $W_{k+1} - W_k$ is a very small value
  - But if $\left(n_{datapoints}\right)$ is large, computing the sum is very expensive. The solution for this stochastic gradient descent algorithm.

- **Stochastic (randomly determined) Gradient descent**

$$\boxed{W_k = W_{k-1} - \eta * \sum_{i=1}^{k} \{-2x_i\}\{y_i - (W_{k-1} * x_i)\}}$$

  - In SGD, instead of taking sum of all n points, we calculate sum of k points where $1 \le k \ge n.$ k is batch size
  - It says pick a random set of k points for each of the iteration and perform gradient descent
  - As we pick k random points, we call this Gradient Descent as Stochastic Gradient Descent;
  - As k reduces from n to 1 the number of iterations required to reach optimum increases; At each iteration we have k and η changing; k is randomly picked between 1 and n which is called as batch size; η is reduced at each iteration progressively. The optimum value reached from both methods will be close.

$$\boxed{W_{GD}^* \approx W_{SGD}^*}$$

- **Constrained Optimization & PCA**
  - General constrained optimization is

$$\boxed{x^* = max_x \ f(x)\ [Objective\ function]\ such\ that\ g(x) = c\ [equality\ constraint],\ h(x) \geq d[inequality\ constraint]}$$

  - We solve such problems by using Lagrangian multipliers λ, μ (both ≥ 0)

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

  - Now we will solve above equation by $\boxed{\frac{\partial \mathcal{L}}{\partial x} = 0, \frac{\partial \mathcal{L}}{\partial \lambda} = 0, \frac{\partial \mathcal{L}}{\partial \mu} = 0}$
  - By solving them we get the values $\tilde{x}, \tilde{\lambda}, \tilde{\mu}$
  - The $x^* = \ max_x \ = \ \tilde{x}$
  - Equation of PCA is $\boxed{max_u \frac{1}{n} \sum_{i=1}^{n} (u^T x_i)^2 \ s.t. \ \ u^T.u = 1}$ here $u^T.u = 1$ is an equality constraint
  - Above equation can also be written as

$$\boxed{max_u u^T.S.u. such\ that \ \ u^T.u = 1}$$

$$\boxed{S = Cov(X) = \frac{X^T.X}{n}}$$

  - S is a covariance vector of X
  - Solving this problem by using Lagrangian multipliers,

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

$$\boxed{\frac{\partial \mathcal{L}}{\partial u} = \frac{\partial}{\partial u} u^T.S.u. - \lambda u^T.u - \ \lambda = 0}$$

$$\boxed{\frac{\partial \mathcal{L}}{\partial u} = \ Su \ - \ \lambda u \ = 0}$$

$$\boxed{Su = \ \lambda u}$$

  - Here S is a co variance matrix. u is the eigen vector of S. & λ is eigen value of S
  - This is because covariance matrix accounts for variability in the dataset, and variability of the dataset is a way to summarize how much information we have in the data (Imagine a variable with all same values as its observations, then the variance is 0, and intuitively speaking, there's not too much information from this variable because every observation is the same). The diagonal elements of the covariance matrix stand for variability of each variable itself, and off-diagonal elements in covariance matrix represents how variables are correlated with each other.

$$\boxed{\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & Cov(z,y) & Cov(z,z) \end{bmatrix}}$$

  - Ultimately, we want our transformed variables to contain as much as information (or equivalently, account for as much variability as possible). PCA performs such orthogonal transformation to convert correlated variables into a new set of uncorrelated variables called principal components. PCA makes sure that the 1st principal component has the largest possible variance, which means the 1st component accounts for the most variability in the data. And that each succeeding component has the highest variance possible under the condition that it is orthogonal to the preceding components.
  - Of course, to account for 100% of the variability, we usually need the same number of components as the original number of variables. But usually with much smaller number of new components, we can explain almost 90% of the variability. Thus, PCA helps us to reduce our focus from a possible large number of variables to a relatively small number of new components, while preserving most of the information from the data.
- **Logistic regression formulation by using Lagrangian multipliers**

$$\boxed{W^* = argmin_x \ (logistic\ loss) + \ \lambda w^T w}$$

  - Above formula is actually a constrained optimization problem

$$\boxed{W^* = argmin_x \ (logistic\ loss)\ such\ that\ w^T w = 1}$$

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

  - For equality constraint $1 - \ w^T w$ or $w^T w - 1$ both are fine.

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

  - In above formula, we can ignore $-\lambda$ as it is a constant. The rest of the equation is same as original one.
  - Hence, one way to understand regularization can be thought as imposing an equality constraint

- **Lagrangian multipliers Example**
  - Find $max_{h,s} 200 h^{\frac{2}{3}} s^{\frac{1}{3}}$ such that $20h + 170s = 20000$
  - $f(x)\ [Objective\ function] =\ max_{h,s} 200 h^{\frac{2}{3}} s^{\frac{1}{3}}$
  - $g(x) = c\ [equality\ constraint] = 20h + 170s = 20000$
  - $\mathcal{L}(x, \lambda) =\ f(x) -\ \lambda\{g(x) - c\}$
  - $\mathcal{L}(h, s, \lambda) =\ 200 h^{\frac{2}{3}} s^{\frac{1}{3}} -\ \lambda(20h + 170s - 20000)$
  - $\mathcal{L}(h, s, \lambda) =\ 200 h^{\frac{2}{3}} s^{\frac{1}{3}} -\ 20h\lambda - 170s\lambda + 20000\lambda$
  - $\frac{\partial \mathcal{L}}{\partial h} = 200 \frac{2}{3} h^{\frac{-1}{3}} s^{\frac{1}{3}} - 20\lambda = 0$
  - $\frac{\partial \mathcal{L}}{\partial s} = 200 \frac{1}{3} h^{\frac{2}{3}} s^{\frac{-2}{3}} - 170\lambda = 0$
  - $\frac{\partial \mathcal{L}}{\partial \lambda} = -\ 20h - 170s + 20000 = 0$
  - If we solve above 3 equations, $h = 666.66, s = 39.12, \lambda = 2.59$

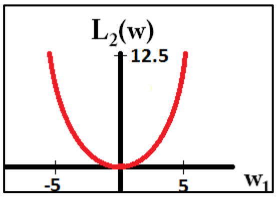
- **Why L1 regularization creates sparsity?**
  - L2 Regularization

$$\boxed{W^* = min_x \cancel{(logistic\ loss)} + \cancel{\lambda}||W||_2^2}$$

$$\boxed{W = [W_1, W_2, \cdots, W_d}$$

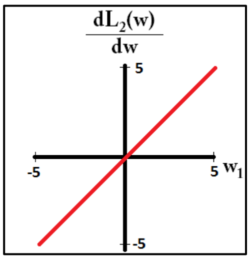
$$\boxed{W^* = min_{w1,w2,\cdots,wd}\ (W_1^2 + W_2^2 + \cdots + W_d^2)}$$

$$\boxed{W^* = min_{w1}\ W_1^2 = \mathcal{L}_2(w_1)}$$

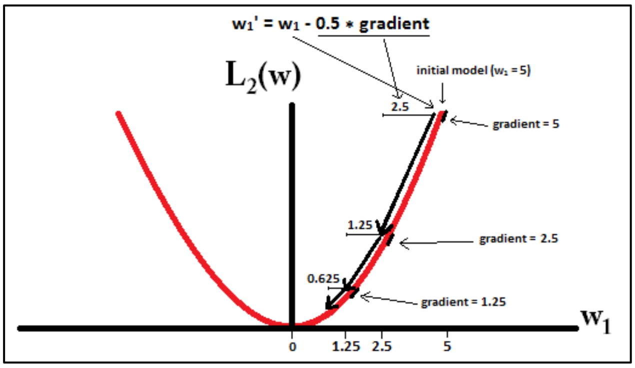


$$\boxed{\mathcal{L}_2(w_1) = W_1^2}$$

$$\boxed{\frac{\partial \mathcal{L}_2}{\partial w_1} = 2 * W_1}$$



By Gradient Descent, Let $(W_1)_j = positive$



$$\boxed{(W_1)_{j+1} = (W_1)_j - \eta \left[\frac{\partial \mathcal{L}_2}{\partial w_1}\right]_{w_{1j}}}$$

$$\boxed{(W_1)_{j+1} = (W_1)_j - \eta(2 * (W_1)_j)}$$

Let$(W_1)_j = 0.1\ and\ \eta = 0.01$

$$\boxed{(W_1)_{j+1} = 0.1 - 0.01(2 * 0.1) = 0.098}$$

- Here, as we come closer and closer towards minima, the slope decreases and the derivative term becomes smaller
- The rate of change is becoming smaller
- By $\mathcal{L}_2$ reg., The value of $(W_1)_{j+1}$ doesn't change much as compared $(W_1)_j$
- $\mathcal{L}_2$ reg. does not change the values much in each iteration
- Here there are less chances of W_1 becoming 0 after k-iterations

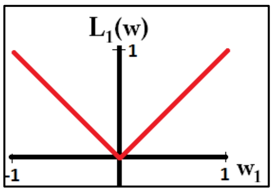
- L1 Regularization

$$\boxed{W^* = min_x \cancel{(logistic\ loss)} + \cancel{\lambda}||W||_1}$$

$$\boxed{W = [W_1, W_2, \cdots, W_d}$$

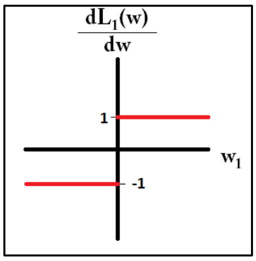
$$\boxed{W^* = min_{w1,w2,\cdots,wd}\ (|W_1| + |W_2| + \cdots + |W_d|)}$$

$$\boxed{W^* = min_{w1}\ |W_1| = \mathcal{L}_1(w_1)}$$

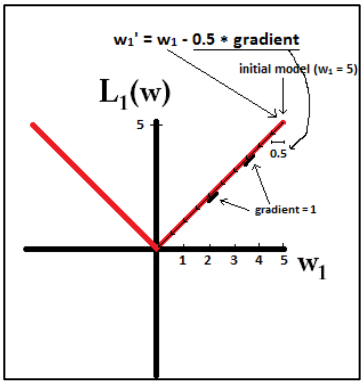


$$\boxed{\mathcal{L}_1(w_1) = |W_1| \begin{matrix} W_1\ if\ W_1 > 0 \\ -(W_1)\ if\ W_1 \le 0 \end{matrix}}$$

$$\boxed{\frac{\partial \mathcal{L}_1}{\partial w_1} = \begin{matrix} +1\ if\ W_1 > 0 \\ -1\ if\ W_1 \le 0 \end{matrix}}$$



By Gradient Descent, Let $(W_1)_j = $ positive



$$\boxed{(W_1)_{j+1} = (W_1)_j - \eta \left[\frac{\partial \mathcal{L}_1}{\partial w_1}\right]_{w_{1j}}}$$

$$\boxed{(W_1)_{j+1} = (W_1)_j - \eta * 1}$$

$Let(W_1)_j = 0.05\ and\ \eta = 0.01$

$$\boxed{(W_1)_{j+1} = 0.1 - 0.01(1) = 0.09}$$

- Here, as we come closer and closer towards minima, the slope remains constant as the derivative term remains constant
- The rate of change remains constant
- By $\mathcal{L}_1$ reg., The value of $(W_1)_{j+1}$ changes a lot as compared $(W_1)_j$
- $\mathcal{L}_1$ reg. continues to constantly reduce W_1 constantly towards $W^* = 0$
- Here there are more chances of W_1 becoming 0 after k- iteration