

# Regression Model

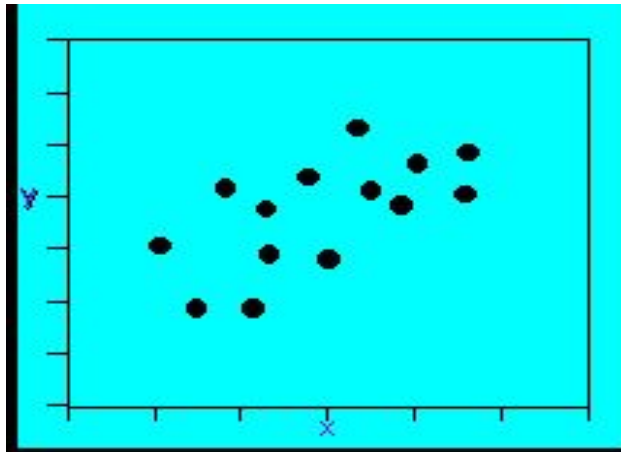


**Jaya Sil**

**Indian Institute of Engineering Science & Technology, Shibpur  
Howrah**

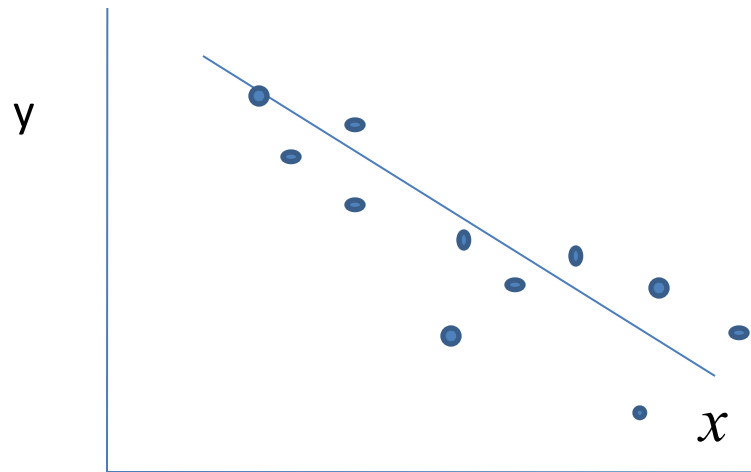
# Linear Model

- **Linear models** describe a continuous response variable as a function of one or more predictor variables.
- Learning a linear relationship between the input attributes (predictor variables) and target values (response variable) values.

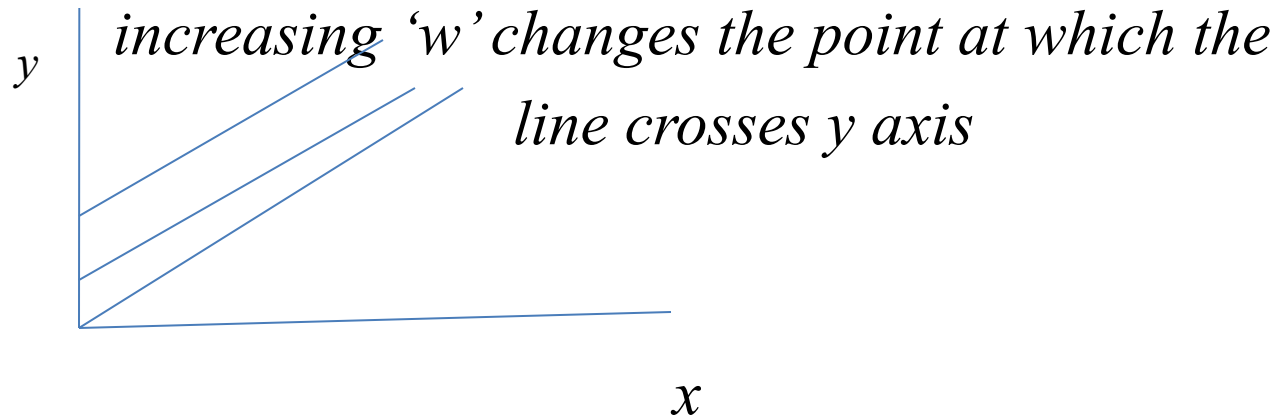


# Linear Model

- Instead of evaluating  $h(x)$  as a function of  $x$ , we make it more flexible using a set of associated parameters.
- $y = wx$  or  $y = h(x; w)$  and the relationship between  $x$  and  $y$  is linear.
- *Assumption: The data could be adequately modeled with a straight line*



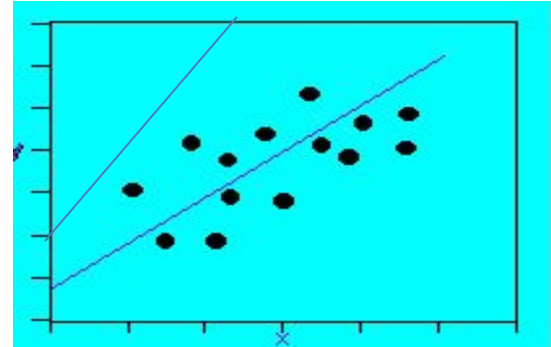
- The assumption is not perfectly satisfied in the Figure.



- *Add a single parameter as  $y = wx$  or  $y = h(x; w)$ ; enhancing the model with any gradient using the choice of  $w$ .*
- But it is not realistic at  $x = 0$  ;  $y = w \times 0$  is zero.
- Adding one more parameter to the model overcomes the problem;  $y = h(x; w_0, w_1) = w_0 + w_1 x$

# Supervised Machine Learning

- Increasing  $w_1$  changes the gradient



- There are many functions which could be used to define the mapping.
- The ultimate goal is to develop a finely tuned predictor function  $h(x)$  such that  $y = h(x)$
- The learning task now involves using the data in figure choose two suitable values of  $w_0$  and  $w_1$

# Supervised Machine Learning

- We decide to approximate  $y$  as a linear function of  $x$ :

$$h(x) = w_0 + w_1 x_1 + w_2 x_2 + \dots + w_n x_n$$

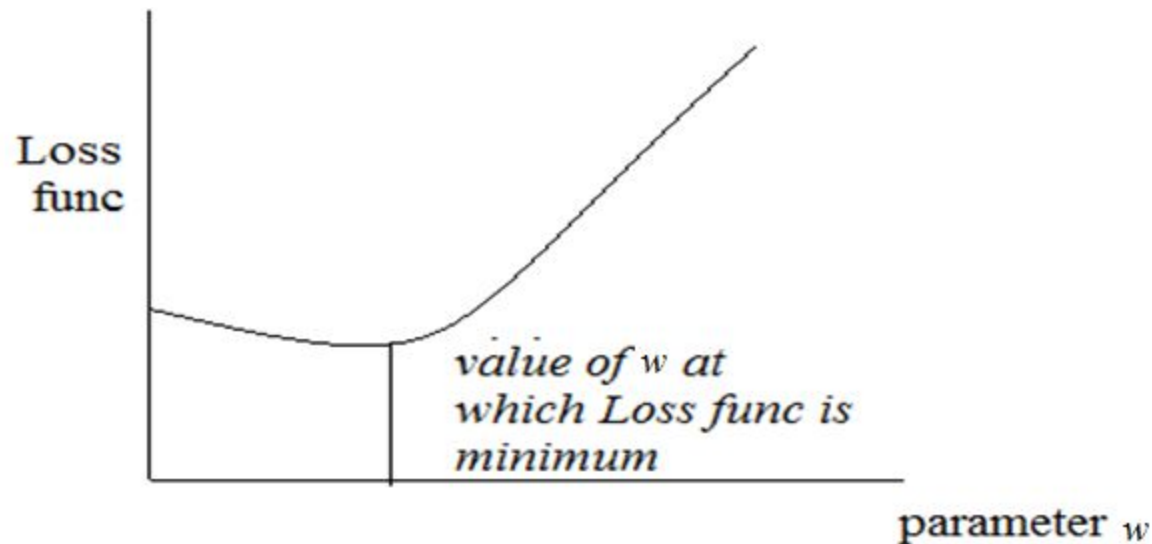
- The  $w_i$ 's are the parameters (also called weights) parameterizing the space of linear functions mapping from  $X$  to  $Y$ .
- simple predictor:  $y = h(x; w_0, w_1) = w_0 + w_1 x$   
Where  $w_0$  and  $w_1$  are constants.
- Our goal is to find the perfect values of  $w_0$  and  $w_1$  in order to make our predictor work as *best* as possible.
- We need to define what is the meaning of *best*.

# Defining a Good Model

- The best solution consists of the values of  $w_0$  and  $w_1$  that produce a line that passes as close as possible to *all* of the data points.
- The minimum squared difference between the target value and the predicted value is a measure of how good is the model.
- The squared difference is defined as:  $(t_n - h(x_n; w_0, w_1))^2$  for  $n$ -th pattern and known as the *squared loss function or cost function*  $L_n()$
- $L_n(t_n, h(x_n; w_0, w_1)) = (t_n - h(x_n; w_0, w_1))^2$

# Loss function

“Learning” optimizes the loss function so that, given input data  $x$  accurately predict value  $h(x)$ .



- Loss is always positive and lower the loss better the function describes the data.

- Average loss function: 
$$L = \frac{1}{N} \sum_{n=1}^N L_n (t_n - h(x_n; w_0, w_1))$$



# Loss Function

- Tune  $w_0$  and  $w_1$  to produce the model that results lowest value of the average Loss function.

- $$L = \underset{w_0, w_1}{arg\ min} \quad \frac{1}{N} \sum_{n=1}^N L_n(t_n - h(x_n; w_0, w_1))$$

- Minimization of the squared loss function is the basis of Least Mean Square Error (LMSE) method of function approximation.
- Other loss functions, like Absolute Loss function

# Gradient Descent Algorithm

- We want to choose  $w$  so as to minimize Loss function.
- Use a search algorithm that starts with some “initial guess” for  $w$ , and that repeatedly changes  $w$  to make Loss smaller.
- Hopefully we converge to a value of  $w$  that minimizes Loss.
- Weight updating:  $w_j := w_j - \eta \frac{\partial L}{\partial w_j}$
- Weight update is simultaneously performed for all values of  $j$ . Here,  $\eta$  is called the learning rate.
- Gradient Descent algorithm repeatedly takes a step in the direction of steepest decrease of  $L$ .

# LMS Algorithm

Tune  $w_0$  and  $w_1$  to produce the model that results lowest value of the average Loss function for a single training pattern.

$$\frac{\partial L}{\partial w_0} = \frac{\partial}{\partial w_0} \frac{1}{2} (t_n - h(x_n; w_0, w_1))^2$$

$$\begin{aligned} \frac{\partial L}{\partial w_0} &= (t_n - h(x_n; w_0, w_1)) \cdot \frac{\partial}{\partial w_0} (t_n - h(x_n; w_0, w_1)) \\ &= -(t_n - h(x_n; w_0, w_1)) \cdot \frac{\partial}{\partial w_0} (\sum_{n=1}^d w_n x_n - t_n) \end{aligned}$$

$$\begin{aligned} \frac{\partial L}{\partial w_1} &= (t_n - h(x_n; w_0, w_1)) \cdot \frac{\partial}{\partial w_1} (t_n - h(x_n; w_0, w_1)) \\ &= -(t_n - h(x_n; w_0, w_1)) \cdot \frac{\partial}{\partial w_1} (\sum_{n=1}^d w_n x_n - t_n) \end{aligned}$$

$$\frac{\partial L}{\partial w_n} = (h(x_n; w_0, w_1) - t_n) x_n$$

# LMS UPDATE RULE

- For a single training example, the update rule is:

$$w_n := w_n - \eta (h(x_n; w_0, w_1) - t_n) x_n$$

- The magnitude of weight updating is proportional to error i.e.

$$(h(x_n; w_0, w_1) - t_n)$$

For N number of training patterns weight update rule:

$$w_n := w_n - \eta \sum_{n=1}^N (h(x_n; w_0, w_1) - t_n) x_n$$

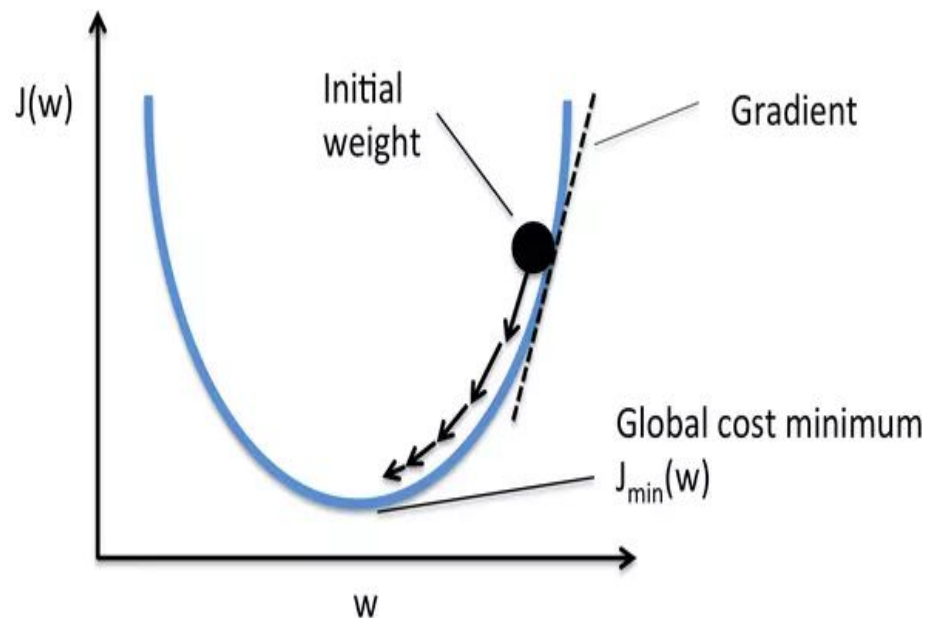
The algorithm will converge when there no weight update takes place in case it is performed iteratively.

# Gradient Descent Algorithm

- The update rule is gradient descent when summation is substituted by  $\frac{\partial L}{\partial w_j}$  i.e. gradient of cost or loss function.
- L is a convex quadratic function, so converges at global minima/maxima.
- When updating is performed for each training example, called Batch Gradient Descent.
- When updating is performed for a set of training example, called Stochastic Gradient descent.

- Searching for points where the gradient of a function is zero, called minima.

To determine the value of the zero gradient point (minima, maxima) we examine the second derivative



# Analytical Solution

$L = 1/N \sum L_n (t_n - h(x_n; w_0, w_1))$ ;  $L$  is average Loss function

$$= 1/N \sum (t_n - h(x_n; w_0, w_1))^2$$

$$= 1/N \sum (t_n - (w_0 + w_1 x_n))^2$$

- Differentiating  $L$  by calculating the partial derivatives with respect to  $w_0$  and  $w_1$  and equating them to zero to obtain  $w_0$  and  $w_1$
- Differentiating again w.r.t.  $w_0$  and  $w_1$  we find the point at which loss is minimum.

# Turning points

- $w_0 = 1/N (\sum t_n) - w_1(1/N(\sum x_n))$  when  $\frac{\partial^2 L}{\partial w_0^2} = 2$
- $w_0^{av} = t^{av} - w_1 x^{av}$

There is one turning point that correspond to minimum loss

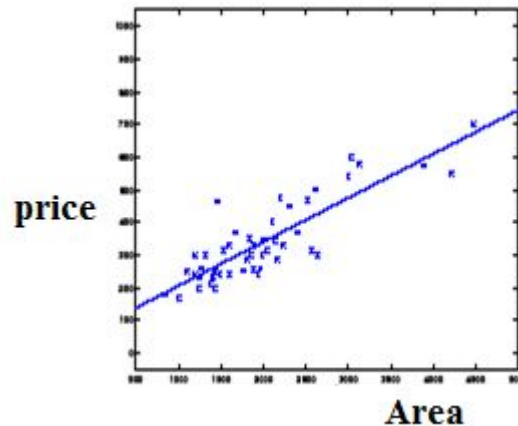
$$w_1^{av} = \frac{\frac{1}{N} (\sum_{n=1}^N x_n t_n) - t^{av} x^{av}}{(\frac{1}{N} \sum_{n=1}^N x_n^2) - x^{av} x^{av}} \quad \text{when} \quad \frac{\partial^2 L}{\partial w_1^2} = \frac{2}{N} \sum_{n=1}^N x_n^2$$

Now we can compute the best parameter values



# Prediction

- Based on linear regression model we predict the output for some input.
- A simple linear model can fit a small dataset and used for prediction.
- $w_0 = 71.27, w_1 = 0.1345$



- Linear model can be extended to larger sets of attributes, modeling complex relationship between input and output.

# Vector-Matrix Notation

- Each data point is described by a set of attributes.
- Solving partial derivatives for each parameter associated with the attributes are time consuming affair.
- Representing attributes of each data point into vector form.
- For example  $n$ -th data point by  $\mathbf{x}_n$  and with two attributes
$$\mathbf{x}_n = [x_{n1}, x_{n2}]^T$$
- Column vectors  $\mathbf{w}$  and  $\mathbf{x}_n$  is defined as  $h(x_n; w_0 w_1) = \mathbf{w}^T \mathbf{x}_n = w_0 + w_1 x_n$

- $L = 1/N \sum (t_n - (w_0 + w_1 x_n))^2 = 1/N \sum (t_n - \mathbf{w}^T \mathbf{x}_n)^2$
- $(\mathbf{t} - \mathbf{X}\mathbf{w})^T(\mathbf{t} - \mathbf{X}\mathbf{w})$  is used to write the loss function.
- $L = 1/N (\mathbf{t} - \mathbf{X}\mathbf{w})^T(\mathbf{t} - \mathbf{X}\mathbf{w})$
- Differentiating loss in vector/matrix form to obtain the vector  $\mathbf{w}$  corresponding to the point where  $L$  is minimum.

$$\frac{\partial L}{\partial \mathbf{w}} = \begin{bmatrix} \frac{\partial L}{\partial w_0} \\ \frac{\partial L}{\partial w_1} \end{bmatrix}$$

# Making Prediction

- Given a new vector of attributes,  $x_{\text{new}}$ , the prediction using the model as  $t_{\text{new}} = \mathbf{W}^T \mathbf{X}_{\text{new}}$
- Linear model of the form with multiple attributes:

$$h(x_1, x_2, \dots, x_n; w_0, w_1, \dots, w_n);$$

$$t_n = w_0 + w_1 x_{n1} + w_2 x_{n2} + \dots + ..$$

- *Prediction from such model is very precise but not always sensible.*

# Learning Task

- Learning using **training examples** : statistically significant random sample.
- If the training set is too small (law of large numbers), we won't learn enough and may even reach inaccurate conclusions.
- For each training example, an input value  $x_{\text{train}}$ , and corresponding output, *y or target* is known in advance.
- For each example, we find the squared difference between the *target*, and predicted value  $h(x_{\text{train}})$ .
- With enough training examples, these differences give us a useful way to measure the “wrongness” of  $h(x)$ .

# Learning Task

- Find parameter values so that the difference makes it “less wrong”.
- This process is repeated over and over until the system has converged on the best values.
- In this way, the predictor becomes trained, and is ready to do some real-world predicting.

# Linear Regression

- Get familiar with objective functions, computing their gradients and optimizing the objectives over a set of parameters.
- Goal is to predict a target value  $y$  using a vector of input values  $x \in \mathbb{R}^n$  where the elements  $x_j$  of  $x$  represent “features” that describe the output  $y$ .
- Suppose many examples of houses where the features for the  $i^{\text{th}}$  house are denoted  $x^{(i)}$  and the price is  $y^{(i)}$ .
- Find a function  $y = h(x)$
- If we succeed in finding a function  $h(x)$  and we have seen enough examples of houses and their prices, we hope that the function  $h(x)$  will also be a good predictor of the house price when we are given the features for a new house where the price is not known.

# Linear Regression

$h_w(x) = \sum_j w_j x_j = w^\top x$ ; functions parametrized by the choice of  $w$ .

- Task is to find  $w$  so that  $h_w(x^{(i)})$  is as close as possible to  $y(i)$ .
- In particular, we search for a  $w$  that minimizes:  
$$L(w) = 1/2 \sum_i (h_w(x^{(i)}) - y^{(i)})^2 = 1/2 \sum_i (w^\top x^{(i)} - y^{(i)})^2$$
- This function is the “cost function” which measures how much error is incurred in predicting  $y^{(i)}$  for a particular choice of  $w$ .
- This may also be called a “loss”, “penalty” or “objective” function.



- Find the choice of  $w$  that minimizes  $L(w)$ .
- The optimization procedure finds the best choice of  $w$
- The gradient  $\nabla_w L(w)$  of a differentiable function  $L$  is a vector that points in the direction of steepest increase as a function of  $w$
- It is easy to see how an optimization algorithm could use this to make a small change to  $w$  that decreases (or increase)  $L(w)$ .

# Optimization Method

- Compute the gradient:
- $\nabla_{\mathbf{w}} L(\mathbf{w}) = \begin{pmatrix} \partial L(\mathbf{w}) / \partial w_1 \\ \partial L(\mathbf{w}) / \partial w_2 \\ \vdots \\ \partial L(\mathbf{w}) / \partial w_n \end{pmatrix}$
- 
- Differentiating the cost function  $L(\mathbf{w})$  with respect to a particular parameter  $w_j$  :
- $\partial L(\mathbf{w}) / \partial w_j = \sum_i x_j^{(i)} (h_{\mathbf{w}}(x^{(i)}) - y^{(i)})$

# Non-Linear Response from a Linear Model

- The linear model in terms of  $w$  and  $x$ :  $h(x; w) = w_0 + w_1x$
- The model is linear in term of  $w$  only:  
 $h(x; w) = w_0 + w_1x + w_2x^2$  but the function is quadratic in terms of data.
- We can add as many power we like to get a polynomial function of any order.
- The general form for a K-th order polynomial:
- $$h(x; \mathbf{w}) = \sum_{k=0}^K w_k x^k \quad OR \quad h(x; w) = w_0 + w_1x^2 + w_2x_1x_2 + w_3x_2^2 + \dots$$

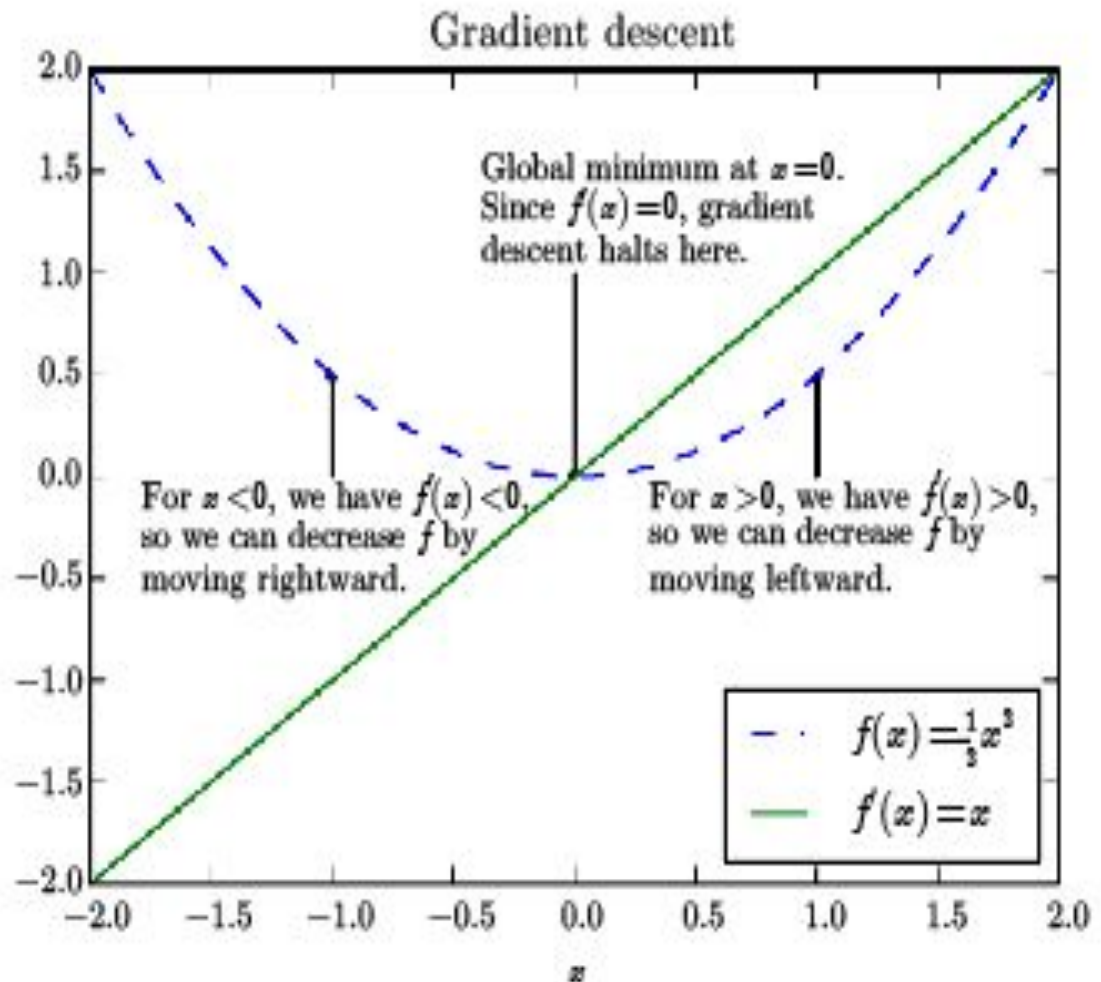
# Gradient Descent

$$y = f(x)$$

$x$  and  $y$  are  
real numbers.

$$dy/dx = f'(x)$$

$f'(x)$  says how to  
change  $x$  for a small  
improvement of  $y$



# Critical Points

When  $dy/dx = f'(x) = 0$ , the derivative provides no information about which direction to move, points are called critical points.

- A local minima is a point where  $y = f(x)$  is lower than at all the neighbouring points. So it is no longer possible to decrease  $f(x)$  by infinitesimal steps.
- A local maxima is a point where  $f(x)$  is higher than neighboring points, so not possible to increase  $f(x)$

Types of critical points

