

The background is a dark blue gradient. It is filled with various light blue and green line-art icons related to technology and machine learning, including gears, circuit boards, a robot, a laptop, a brain, a globe, and a book. The words "MACHINE LEARNING" are written in large, light blue, outlined capital letters across the center. Overlaid on this is a white rectangular frame with a thin border. Inside this frame, the words "Linear Regression" are written in a large, white, sans-serif font.

Linear Regression

Why Linear Regression ?



Linear Regression... What's new?



Features

The features are the elements of your input vectors. The number of features is equal to the number of nodes in the input layer of the network

Category	Features
Housing Prices	No. of Rooms, House Area, Air Pollution, Distance from facilities, Economic Index city, Security Ranking etc.
Spam Detection	presence or absence of certain email headers, the email structure, the language, the frequency of specific terms, the grammatical correctness of the text etc.
Speech Recognition	noise ratios, length of sounds, relative power of sounds, filter matches
Cancer Detection	Clump thickness, Uniformity of cell size, Uniformity of cell shape, Marginal adhesion, Single epithelial cell size, Number of bare nuclei, Bland chromatin, Number of normal nuclei, Mitosis etc.
Cyber Attacks	IP address, Timings, Location, Type of communication, traffic details etc.
Video Recommendations	Text matches, Ranking of the video, Interest overlap, history of seen videos, browsing patterns etc.
Image Classification	Pixel values, Curves, Edges etc.

Weights

Weights correspond to each feature.

Weights denote how much the feature matters in the model.

Higher weight of a particular feature means that it is more important in deciding the outcome of the model.

Weights of a feature represent that how much evidence it gives in favor or against the current hypothesis in context of the existence or non-existence of the pattern you are trying to identify in the current input.

Generally weights are initialized randomly.

we try to bring them to near optimal values so that they are able to fit the model well and can help in prediction of unseen values

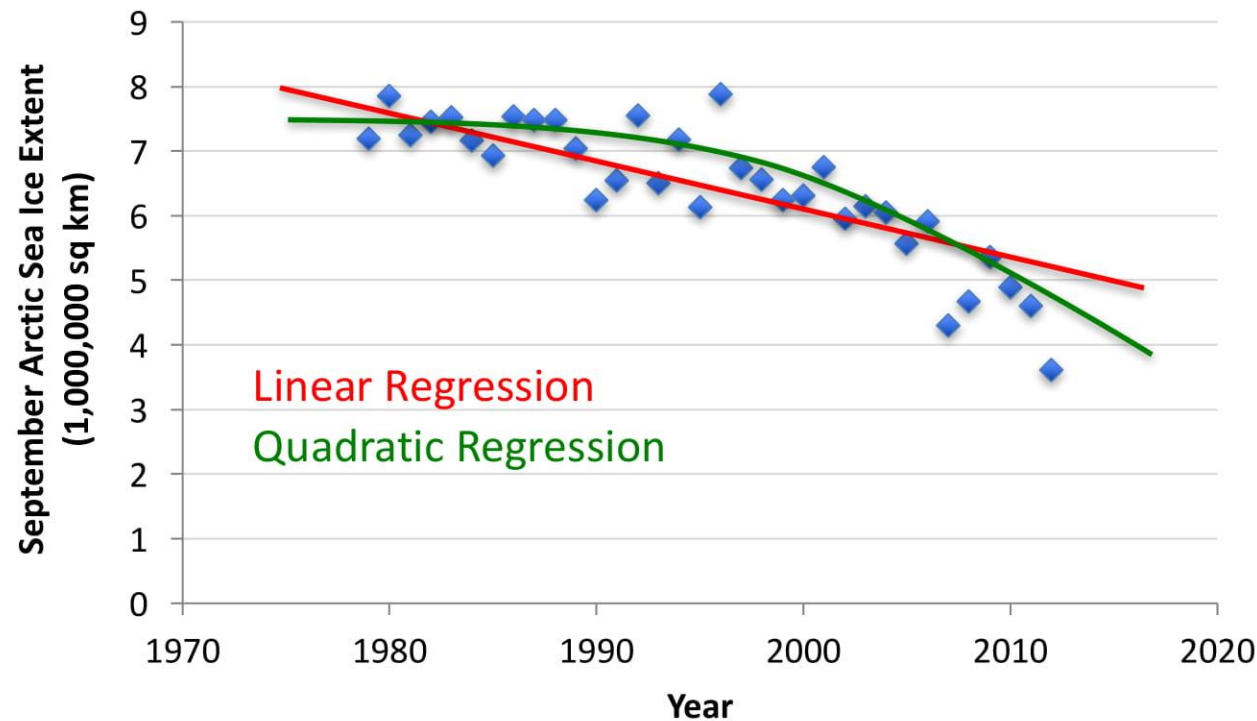
Linear Regression

Linear Regression: For applications where output will be a real value e.g. Predicting housing price, or predicting price of a share in stock market. In most cases we have multiple dependent variables, and we call it multiple linear regression

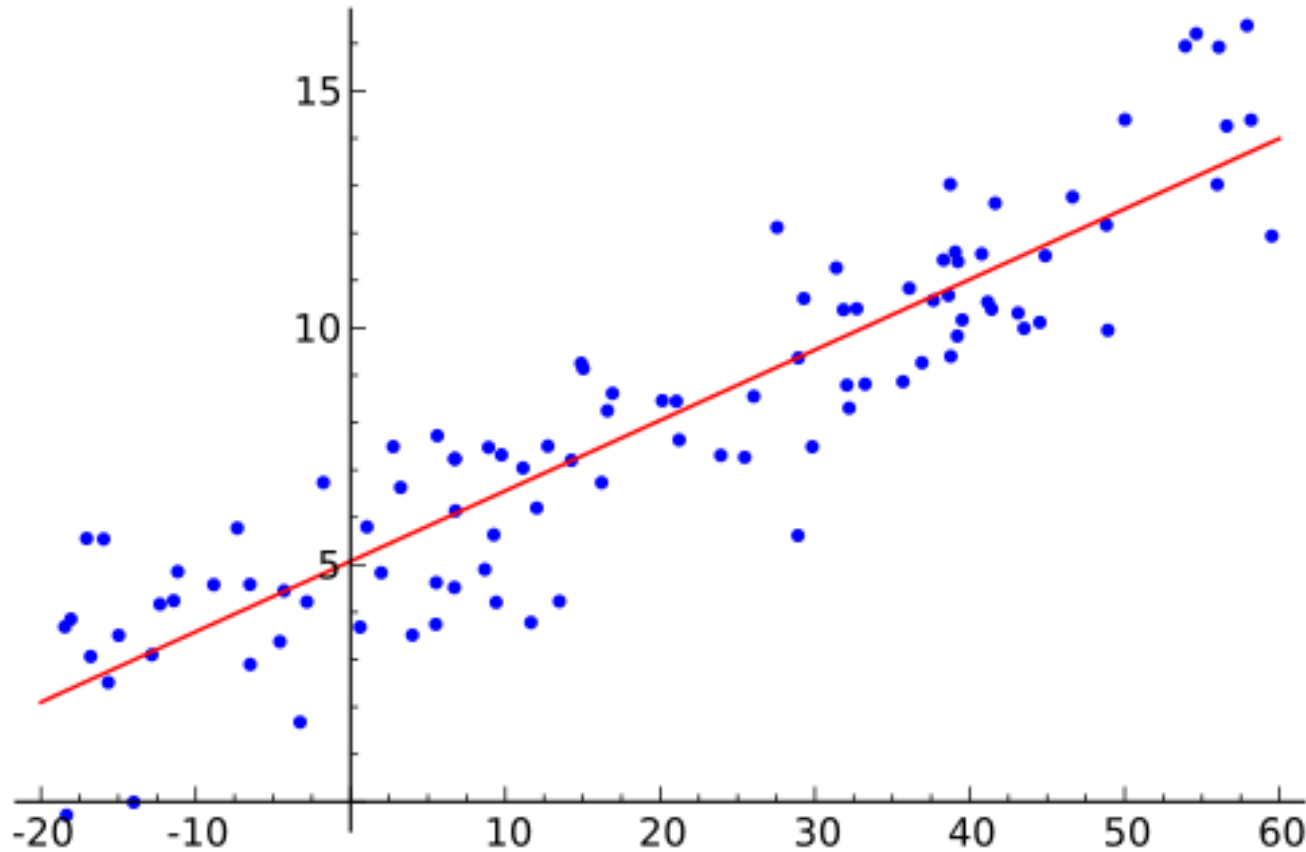
Regression

Given:

- Data $\mathbf{X} = \{x^{(1)}, \dots, x^{(n)}\}$ where $x^{(i)} \in \mathbb{R}^d$
- Corresponding labels $\mathbf{y} = \{y^{(1)}, \dots, y^{(n)}\}$ where $y^{(i)} \in \mathbb{R}$



Linear Regression



$$y = \Theta x + b$$

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^m (\hat{Y}_i - Y_i)^2$$

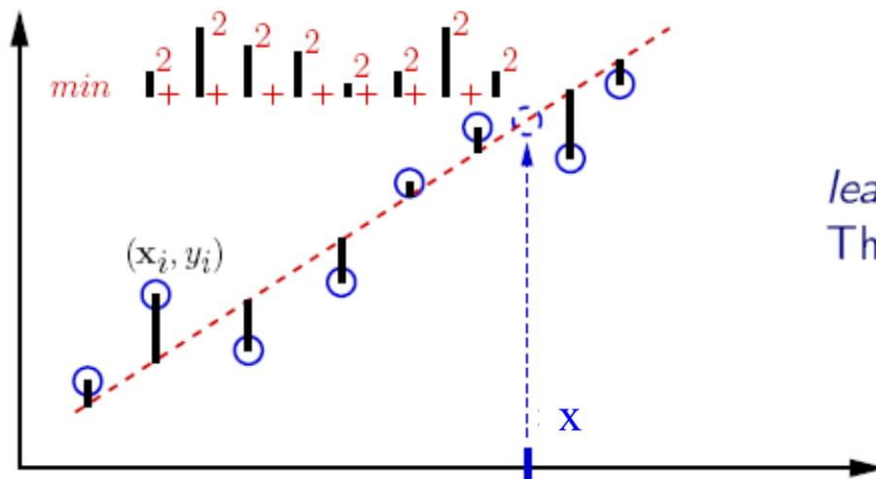
Linear Regression

- Hypothesis:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d = \sum_{j=0}^d \theta_j x_j$$

Assume $x_0 = 1$

- Fit model by minimizing sum of squared errors



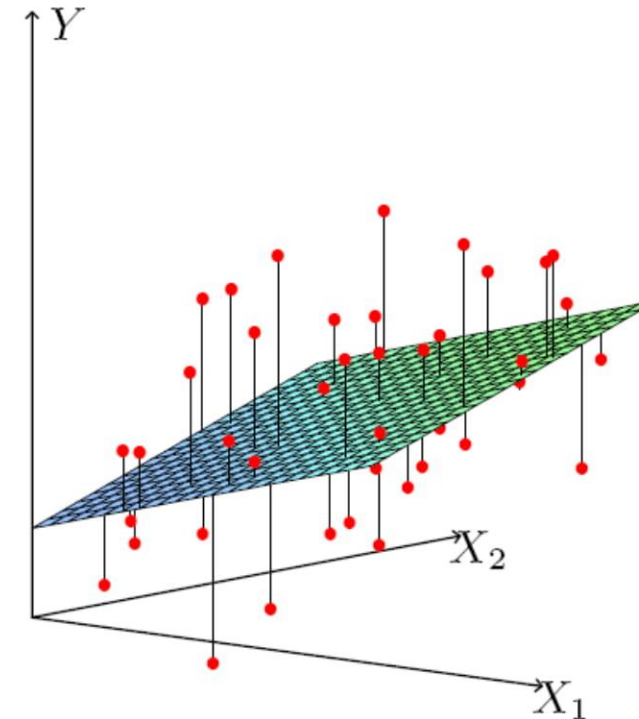
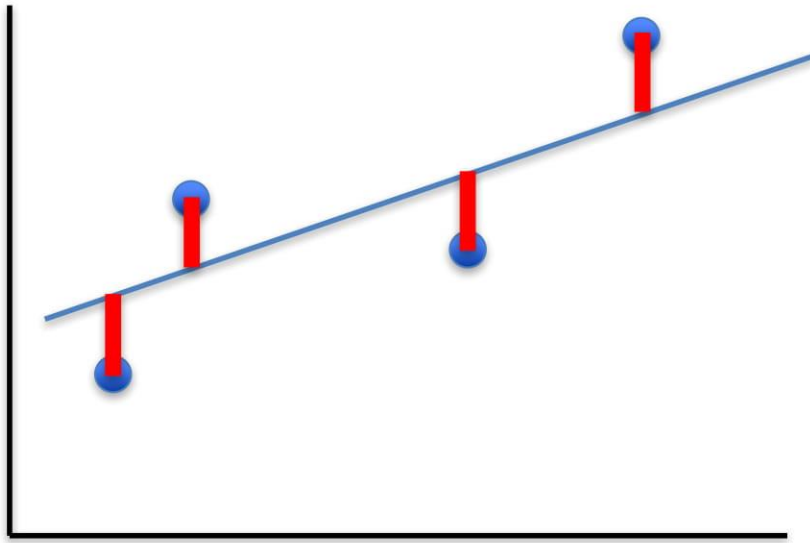
Errors are called “Residuals”

Least Squares Linear Regression

- Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}} \left(\mathbf{x}^{(i)} \right) - y^{(i)} \right)^2$$

- Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$



Loss Function Squaring the Error difference

Benefits of squaring


Squaring always gives a positive value, so the sum will not be zero.

Squaring emphasizes larger differences—a feature that turns out to be both good and bad (think of the effect outliers have).

How to learn proper θ using $J(\theta)$?

- Hypothesis:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_d x_d = \sum_{j=0}^d \theta_j x_j$$

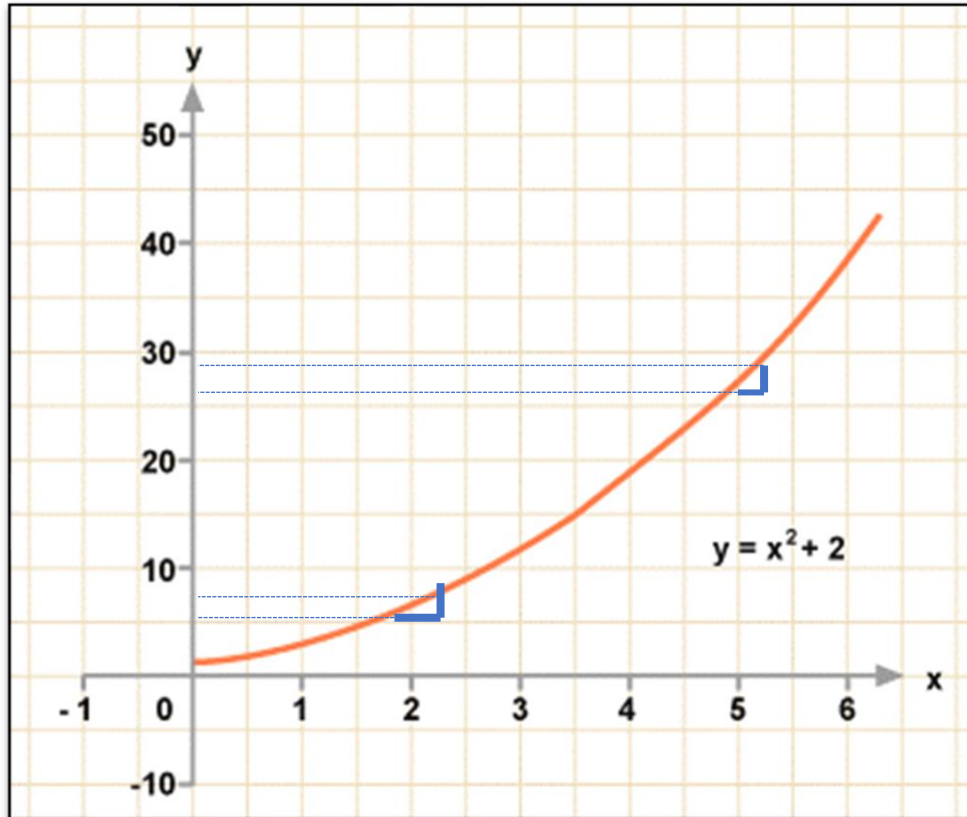
 Assume $x_0 = 1$

Cost Function

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^n \left(h_{\boldsymbol{\theta}} \left(\mathbf{x}^{(i)} \right) - y^{(i)} \right)^2$$

Fit by solving $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$

Derivatives



$$y=f(x)=x^2+2$$

$$\frac{df(x)}{dx} = 2x$$

$$x=2 \quad f(x)=6$$

$$x=2.0001 \quad f(x)=6.00040001$$

$$\text{Slope at } x=2 \text{ is } .0004/.0001 = 4$$

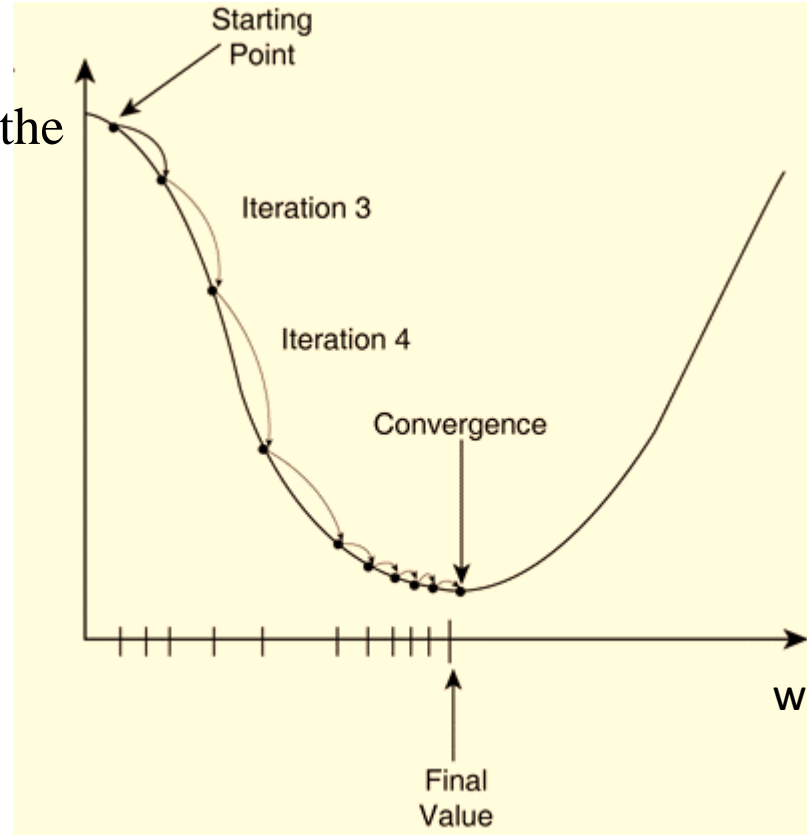
$$x=5 \quad f(x) = 27$$

$$x=5.0001 \quad f(x) = 27.00100001$$

$$\text{Slope at } a = 5 \text{ is } .0010/.0001 = 10$$

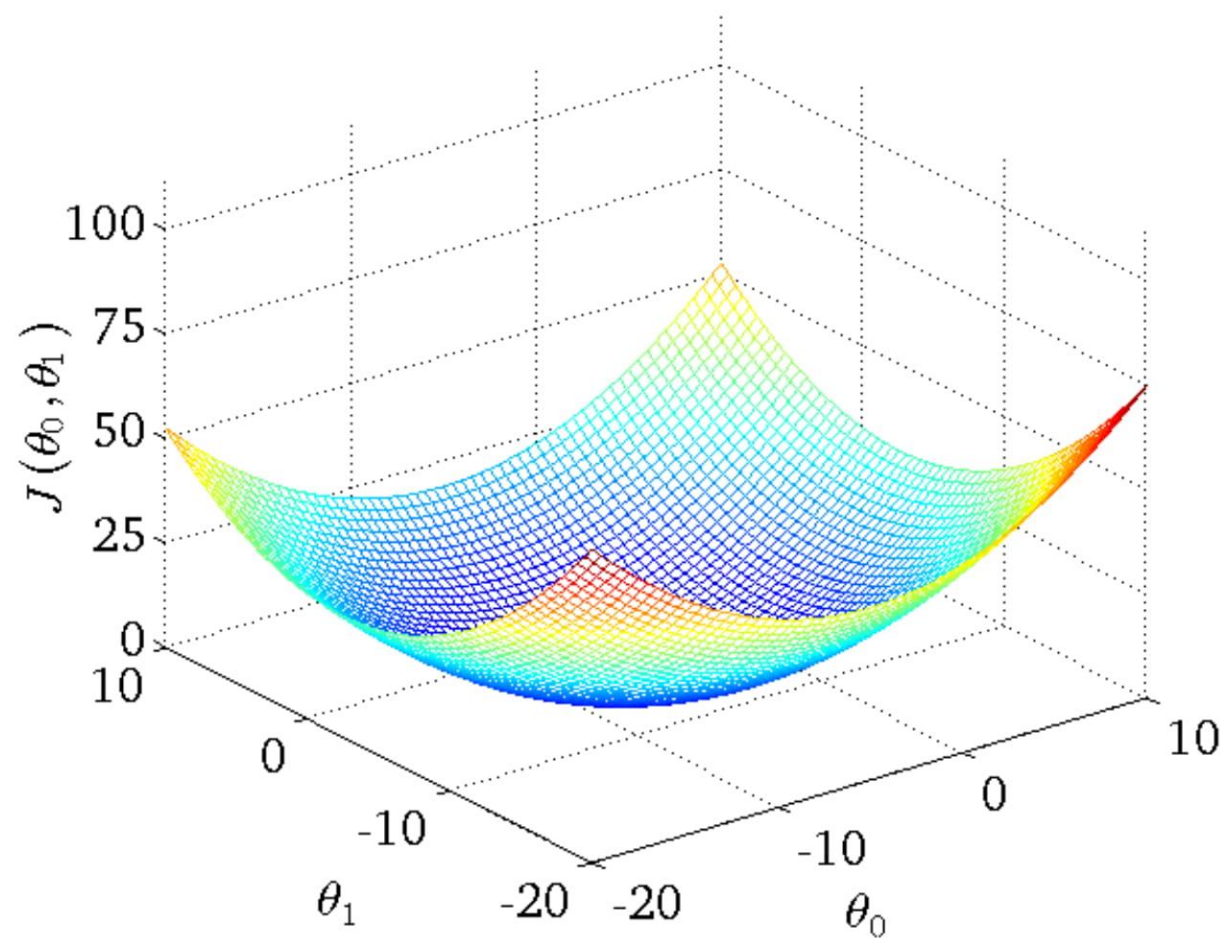
Gradient Descent single dimension

$\frac{dj(w)}{dw}$ is less than the optimum value



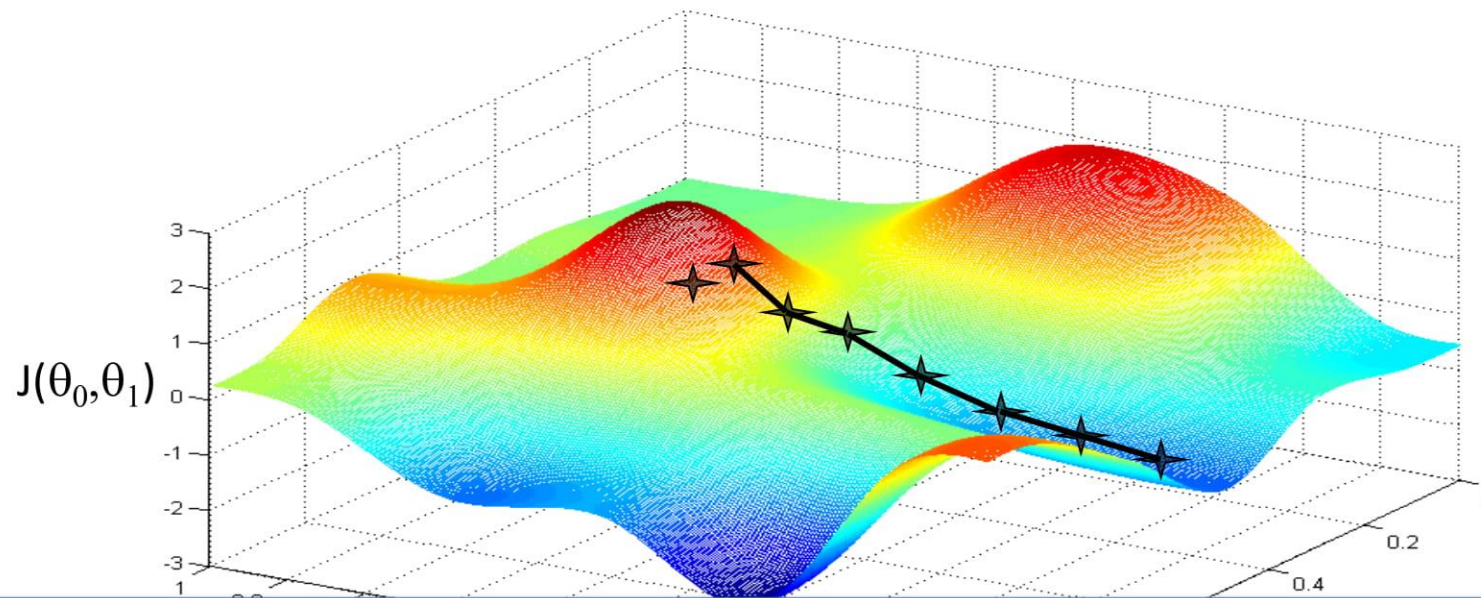
$\frac{dj(w)}{dw}$ is more than the optimum value

Intuition Behind Cost Function



Basic Search Procedure

- Choose initial value for θ
- Until we reach a minimum:
 - Choose a new value for θ to reduce $J(\theta)$



Since the least squares objective function is convex (concave), we don't need to worry about local minima in linear regression

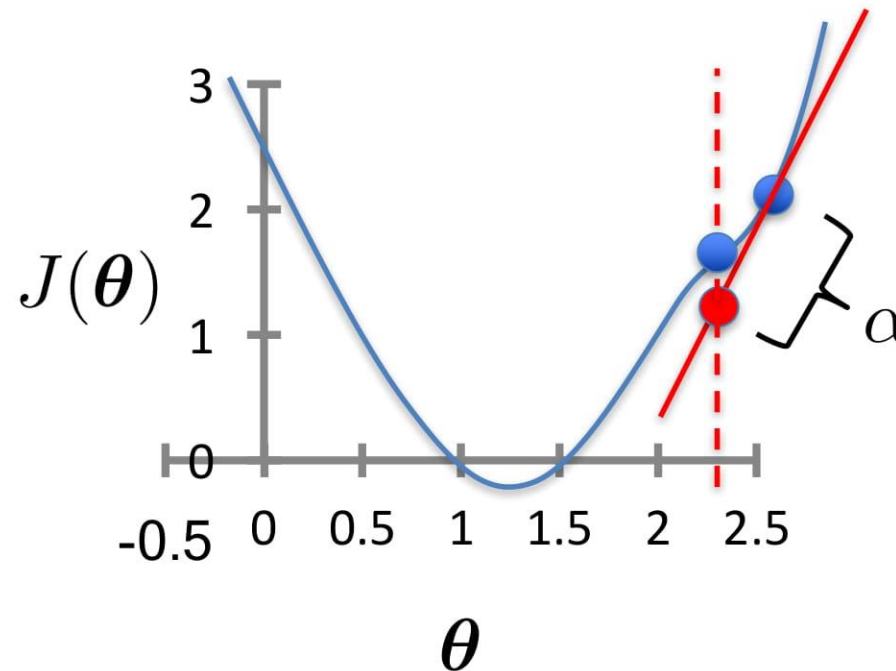
Gradient Descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

simultaneous update
for $j = 0 \dots d$

learning rate (small)
e.g., $\alpha = 0.05$



Gradient Descent

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

simultaneous update
for $j = 0 \dots d$

For Linear Regression:

$$\begin{aligned} \frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left(h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 \\ &= \frac{\partial}{\partial \theta_j} \frac{1}{2n} \sum_{i=1}^n \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right)^2 \\ &= \frac{1}{n} \sum_{i=1}^n \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) \times \frac{\partial}{\partial \theta_j} \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) \\ &= \frac{1}{n} \sum_{i=1}^n \left(\sum_{k=0}^d \theta_k x_k^{(i)} - y^{(i)} \right) x_j^{(i)} \end{aligned}$$

Gradient Descent for Linear Regression

- Initialize θ
- Repeat until convergence

$$\theta_j \leftarrow \theta_j - \alpha \frac{1}{n} \sum_{i=1}^n \left(h_{\theta} \left(\mathbf{x}^{(i)} \right) - y^{(i)} \right) x_j^{(i)} \quad \begin{array}{l} \text{simultaneous} \\ \text{update} \\ \text{for } j = 0 \dots d \end{array}$$

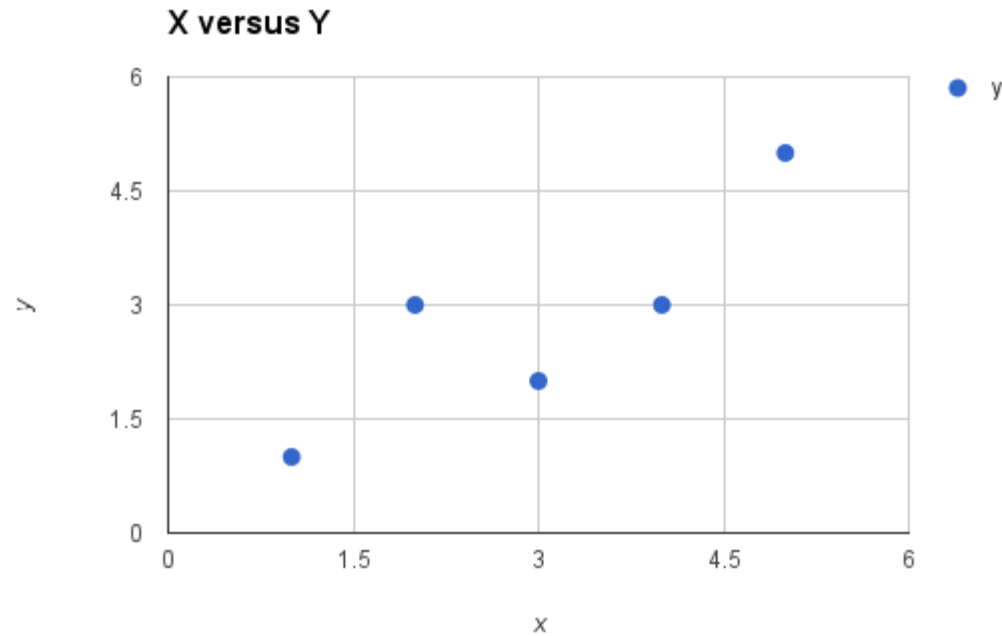
- To achieve simultaneous update
 - At the start of each GD iteration, compute $h_{\theta} \left(\mathbf{x}^{(i)} \right)$
 - Use this stored value in the update step loop
- Assume convergence when $\|\theta_{new} - \theta_{old}\|_2 < \epsilon$

Gradient Descent Numerical Example

Data

1	x	y
2	1	1
3	2	3
4	4	3
5	3	2
6	5	5

Plot



Linear Regression Model

$$y = \Theta_0 + \Theta_1 * x$$

Gradient Descent Numerical Example

- Update rule **$\Theta = \Theta - \text{alpha} * \text{delta}$**
- For 1 sample **delta** or gradient is just error
- Θ – parameter
- alpha – learning rate
- delta – Gradient with respect to the Θ

Model

$$p = \Theta_0 + \Theta_1 * x$$

Error

$$\text{error} = p(i) - y(i)$$

Gradient Descent Numerical Example

Initial Model

$$\Theta_0 = 0.0$$

$$\Theta_1 = 0.0$$

$$p = 0.0 + 0.0 * x$$

Iteration 1

$$x=1, y=1$$

$$p = 0.0 + 0.0 * 1$$

$$p = 0$$

$$\text{error} = 0 - 1$$

$$\text{error} = -1$$

Gradient Descent Numerical Example

Parameter Updation

$$\Theta_0(t+1) = \Theta_0(t) - \text{alpha} * \text{error}$$

$$\Theta_0(t+1) = 0.0 - 0.01 * -1.0$$

$$\Theta_0(t+1) = 0.01 \longleftarrow \text{Updated Bias}$$

$$\Theta_1(t+1) = \Theta_1(t) - \text{alpha} * \text{error} * x$$

$$\Theta_1(t+1) = 0.0 - 0.01 * -1 * 1$$

$$\Theta_1(t+1) = 0.01 \longleftarrow \text{Updated Weight/parameter}$$

Gradient Descent Numerical Example

Further Iterations

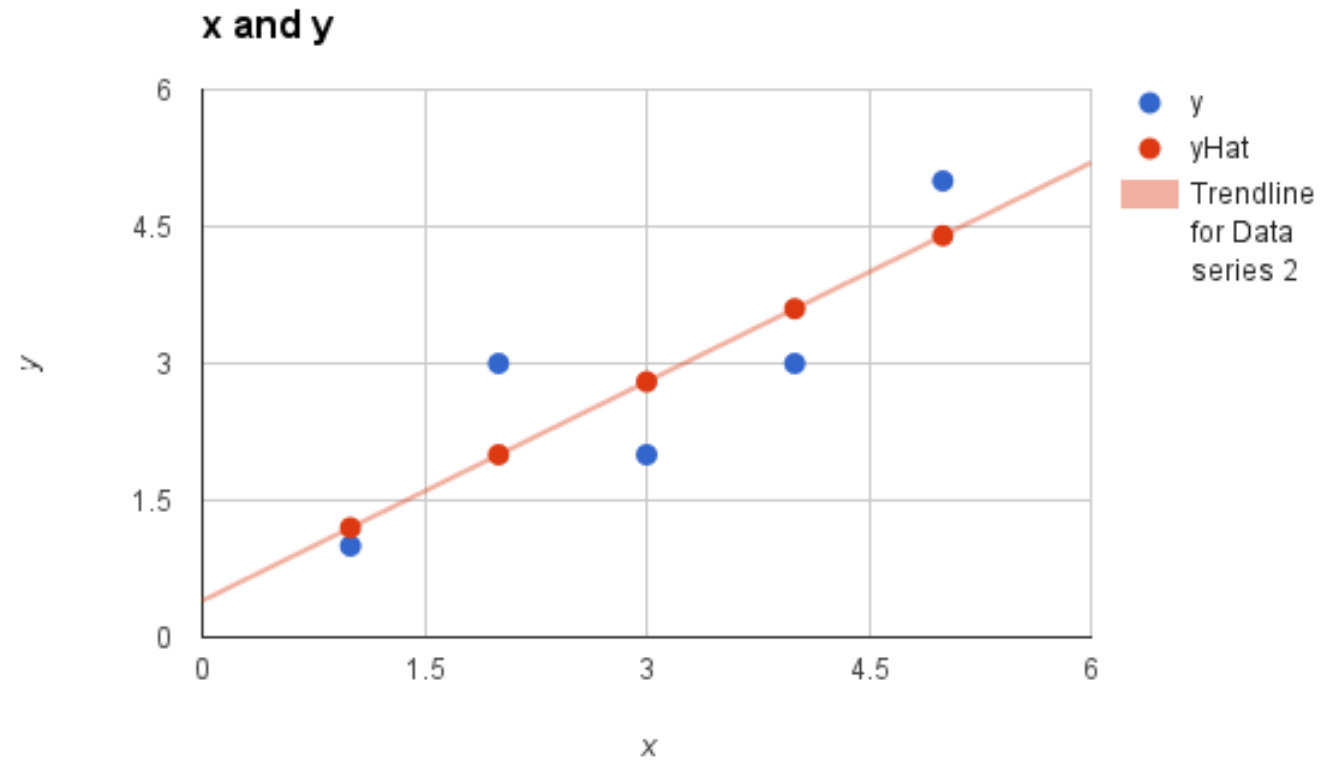
Iteration	θ_0	θ_1
1	0.01	0.01
2	0.0397	0.0694
3	0.066527	0.176708
4	0.08056	0.218808
5	0.118814	0.410078
6	0.123526	0.414789
7	0.143994	0.455727
8	0.154325	0.497051
9	0.157871	0.507687
10	0.180908	0.622872
11	0.18287	0.624834
12	0.198544	0.656183
13	0.200312	0.663252
14	0.198411	0.65755
15	0.213549	0.733242
16	0.214081	0.733774
17	0.227265	0.760141
18	0.224587	0.749428
19	0.219858	0.735242
20	0.230897	0.790439



Gradient Descent Numerical Example

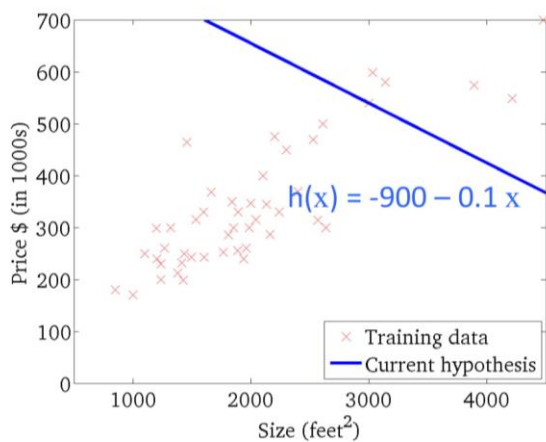
Prediction

x	y	prediction
1	1	0.9551
2	3	1.690342
4	3	3.160826
3	2	2.425584
5	5	3.896068



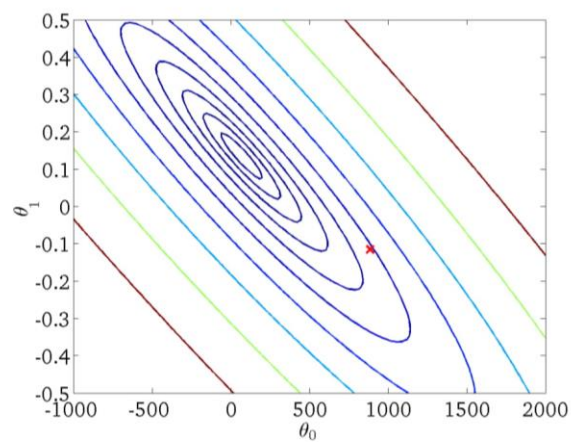
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



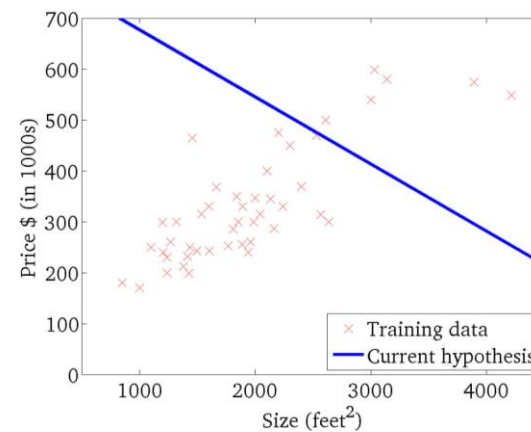
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



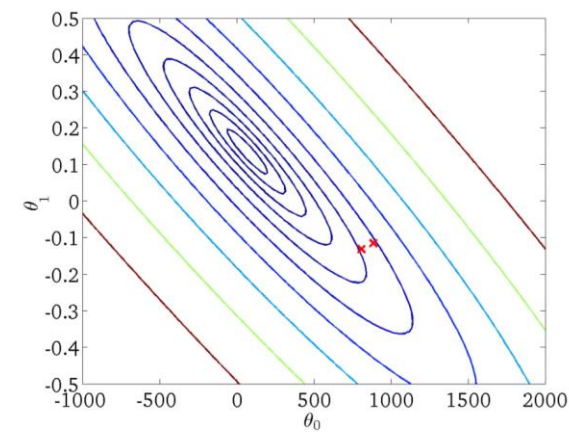
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



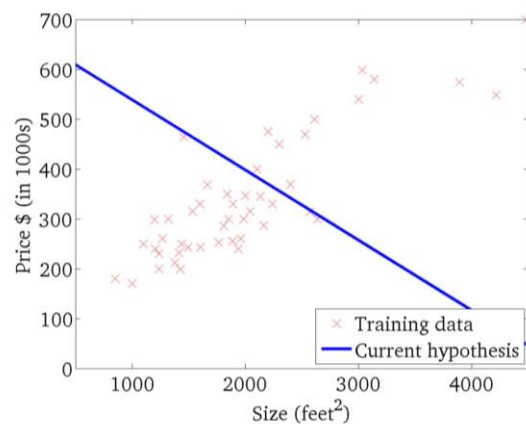
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



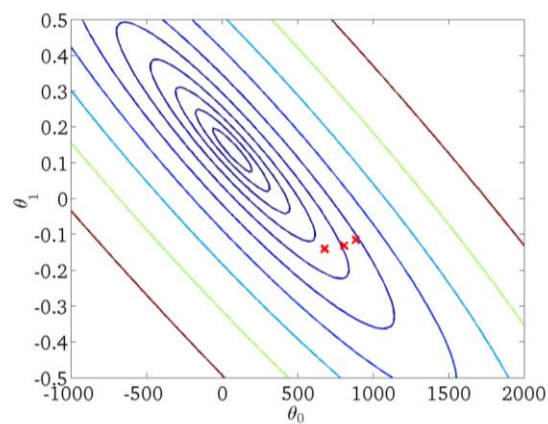
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



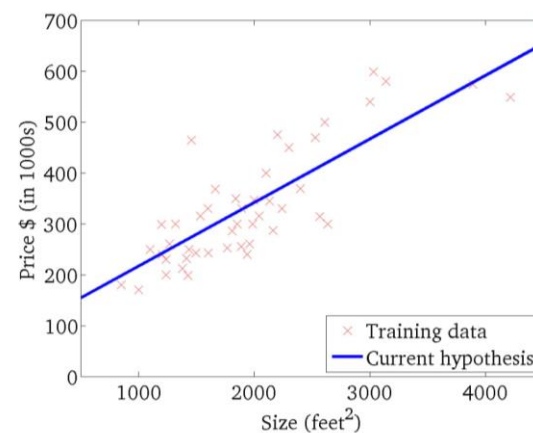
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)



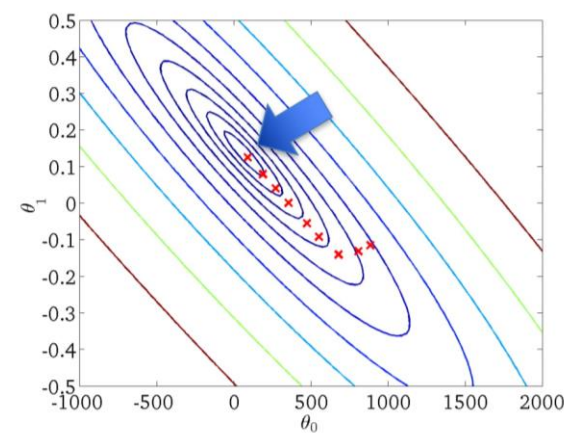
$$h_{\theta}(x)$$

(for fixed θ_0, θ_1 , this is a function of x)



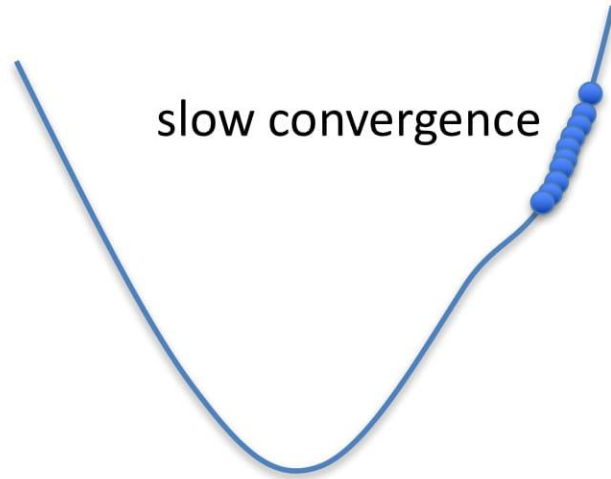
$$J(\theta_0, \theta_1)$$

(function of the parameters θ_0, θ_1)

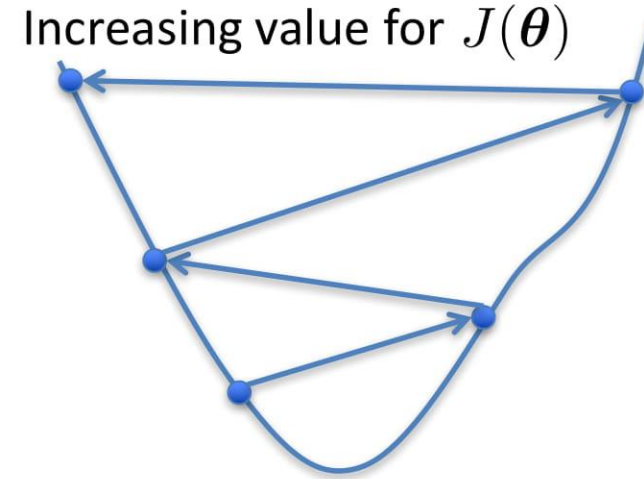


Choosing α

α too small



α too large

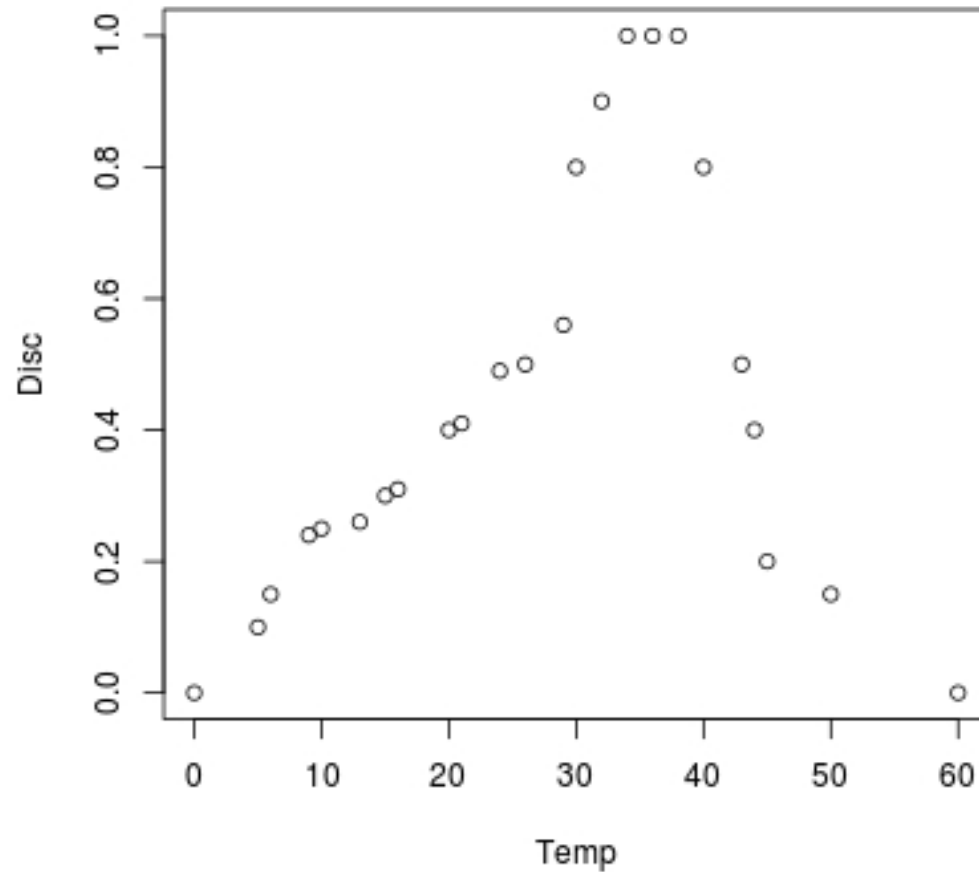


- May overshoot the minimum
- May fail to converge
- May even diverge

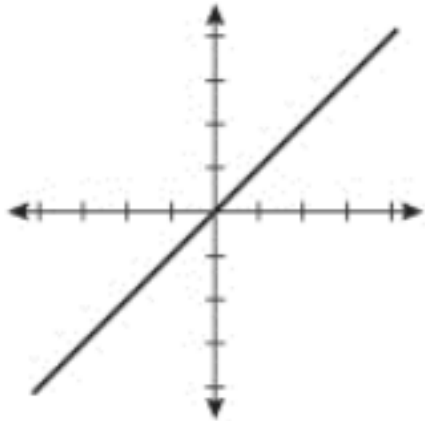
To see if gradient descent is working, print out $J(\theta)$ each iteration

- The value should decrease at each iteration
- If it doesn't, adjust α

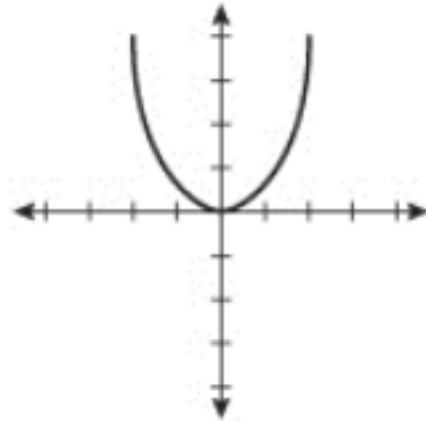
What if data is non-linear?



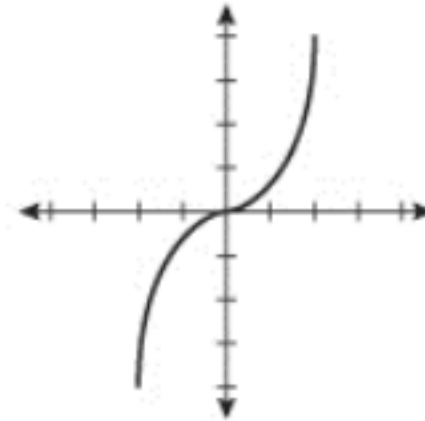
Data Transformation



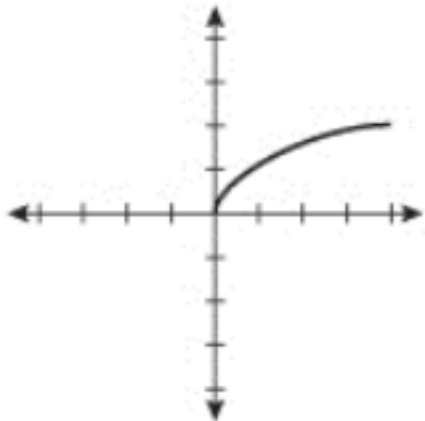
$$y = f(x) = x$$



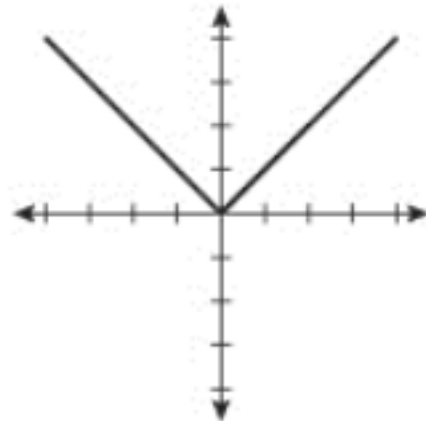
$$y = f(x) = x^2$$



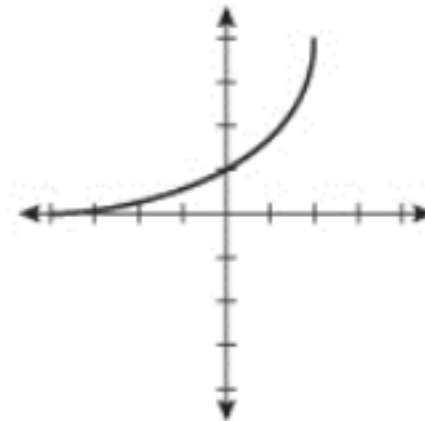
$$y = f(x) = x^3$$



$$y = f(x) = \sqrt{x}$$



$$y = f(x) = |x|$$



$$y = f(x) = 2^x$$

Extending Linear Regression to More Complex Models

- The inputs \mathbf{X} for linear regression can be:
 - Original quantitative inputs
 - Transformation of quantitative inputs
 - e.g. log, exp, square root, square, etc.
 - Polynomial transformation
 - example: $y = \beta_0 + \beta_1 \cdot x + \beta_2 \cdot x^2 + \beta_3 \cdot x^3$
 - Basis expansions
 - Dummy coding of categorical inputs
 - Interactions between variables
 - example: $x_3 = x_1 \cdot x_2$

This allows use of **linear** regression techniques to fit **non-linear** datasets.

Linear Basis Function Models

- Generally,

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{j=0}^d \theta_j \underbrace{\phi_j(\boldsymbol{x})}_{\text{basis function}}$$

- Typically, $\phi_0(\boldsymbol{x}) = 1$ so that θ_0 acts as a bias
- In the simplest case, we use linear basis functions :

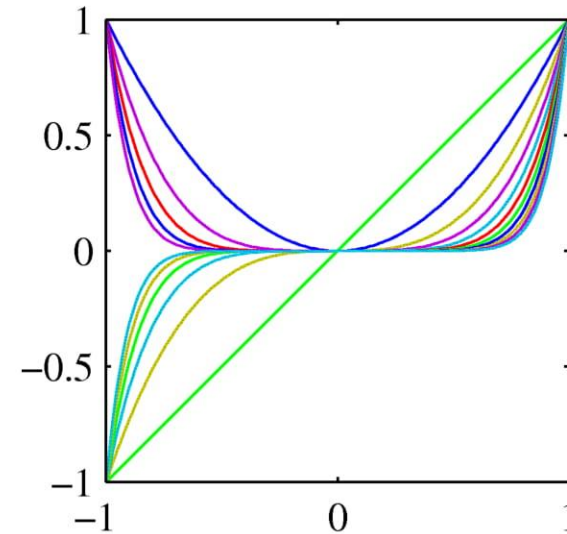
$$\phi_j(\boldsymbol{x}) = x_j$$

Linear Basis Function Models

- Polynomial basis functions:

$$\phi_j(x) = x^j$$

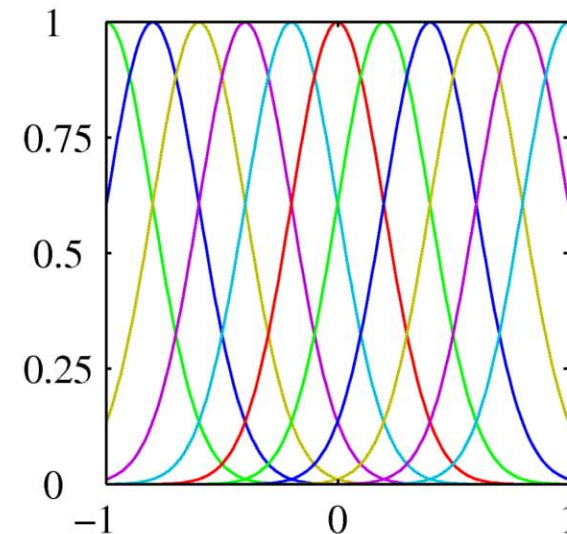
- These are global; a small change in x affects all basis functions



- Gaussian basis functions:

$$\phi_j(x) = \exp \left\{ -\frac{(x - \mu_j)^2}{2s^2} \right\}$$

- These are local; a small change in x only affect nearby basis functions. μ_j and s control location and scale (width).



Linear Basis Function Models

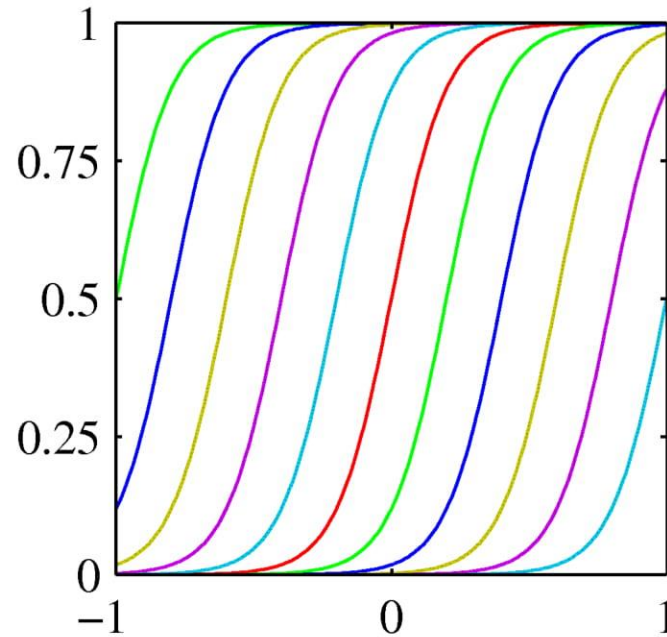
- Sigmoidal basis functions:

$$\phi_j(x) = \sigma \left(\frac{x - \mu_j}{s} \right)$$

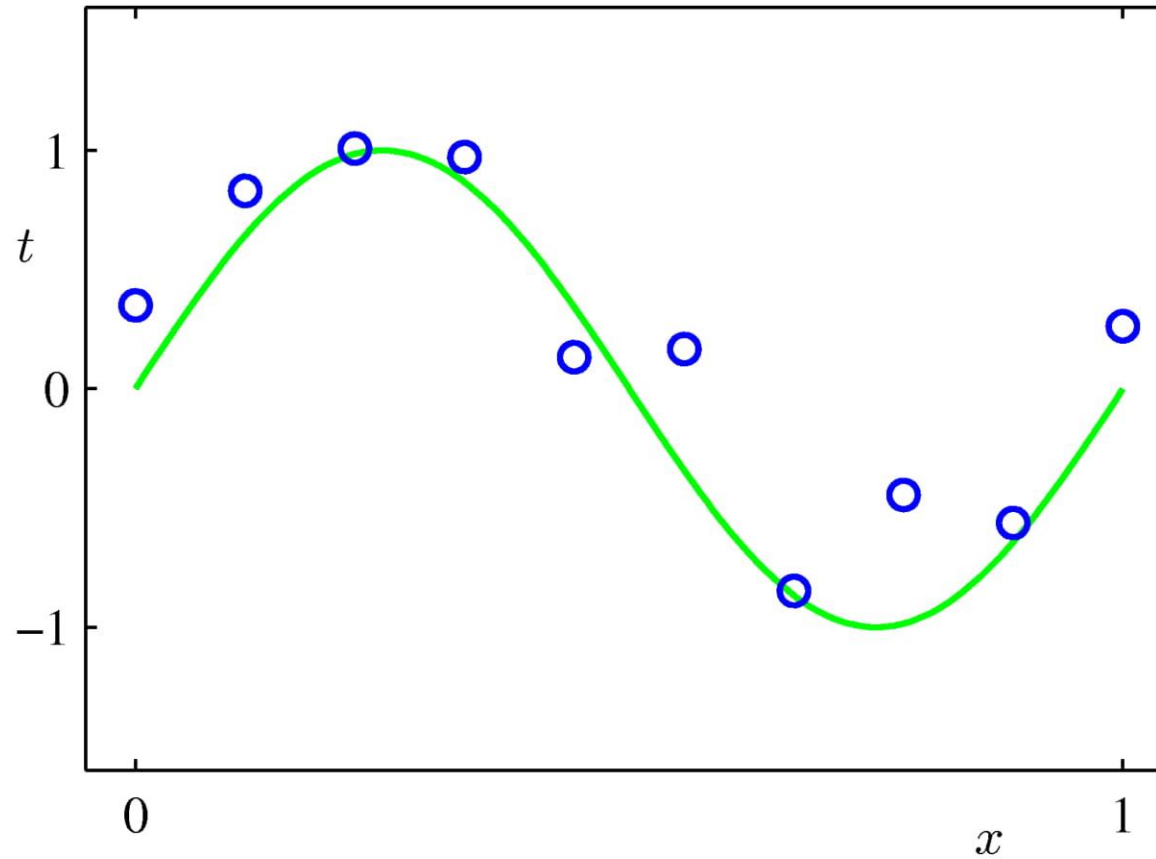
where

$$\sigma(a) = \frac{1}{1 + \exp(-a)}$$

- These are also local; a small change in x only affects nearby basis functions. μ_j and s control location and scale (slope).



Example of Fitting a Polynomial Curve with a Linear Model



$$y = \theta_0 + \theta_1 x + \theta_2 x^2 + \dots + \theta_p x^p = \sum_{j=0}^p \theta_j x^j$$

Links for Demo

Fitting a line:

<https://www.geogebra.org/m/FUe3HfRf>

Polynomial Regression

<https://arachnoid.com/polysolve/>

