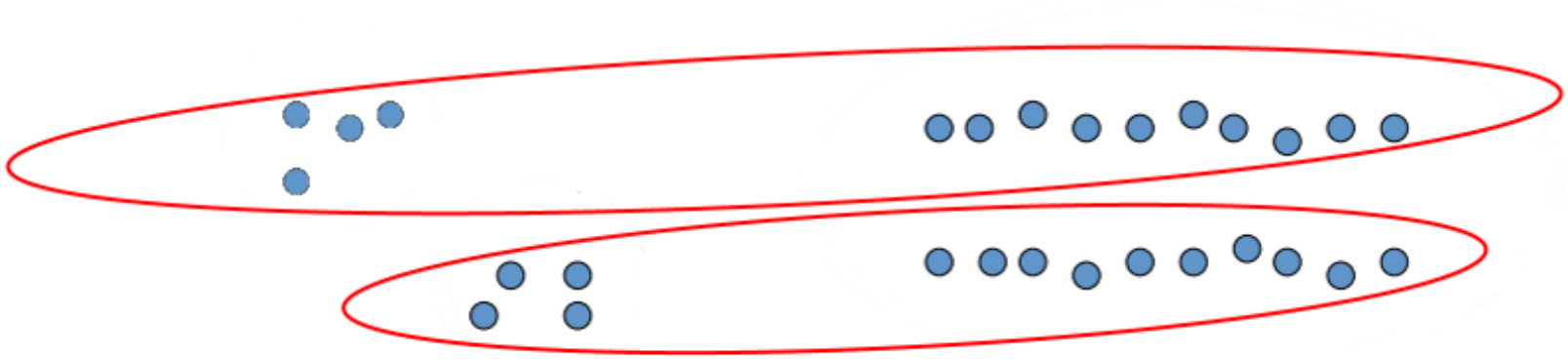
Clustering – An example



Content

Content

Classification - an example

A power generator

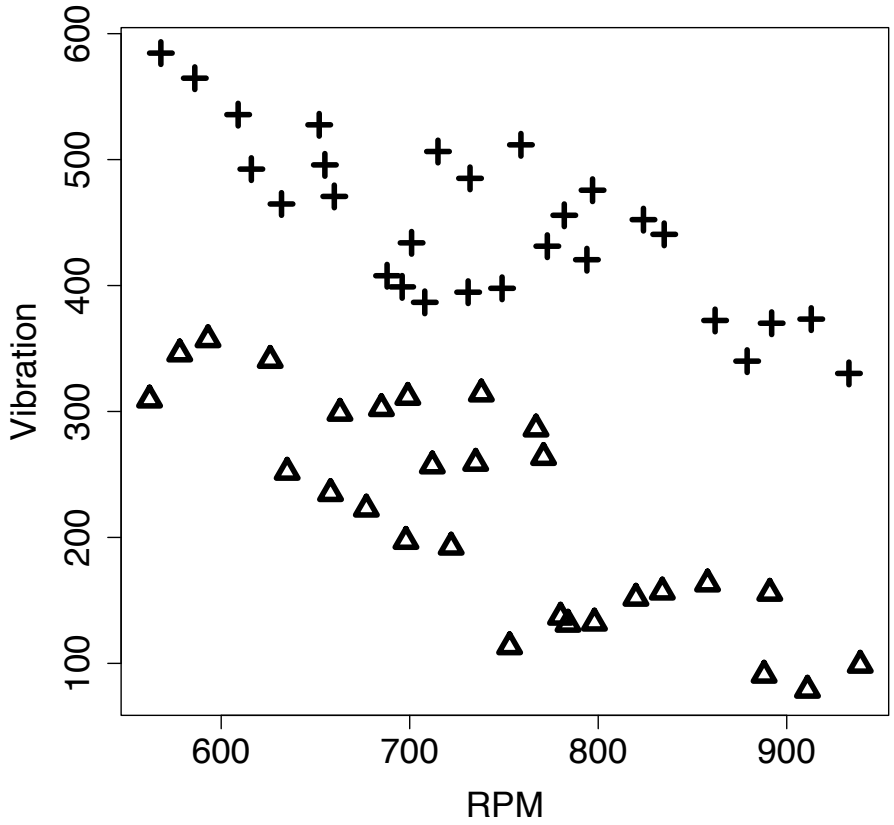
Table: A dataset listing features for a number of generators.

ID	RPM	VIBRATION	STATUS
1	568	585	good
2	586	565	good
3	609	536	good
4	616	492	good
5	632	465	good
6	652	528	good
7	655	496	good
8	660	471	good
9	688	408	good
10	696	399	good
11	708	387	good
12	701	434	good
13	715	506	good
14	732	485	good
15	731	395	good
16	749	398	good
17	759	512	good
18	773	431	good
19	782	456	good
20	797	476	good
21	794	421	good
22	824	452	good
23	835	441	good
24	862	372	good
25	879	340	good
26	892	370	good
27	913	373	good
28	933	330	good

ID	RPM	VIBRATION	STATUS
29	562	309	faulty
30	578	346	faulty
31	593	357	faulty
32	626	341	faulty
33	635	252	faulty
34	658	235	faulty
35	663	299	faulty
36	677	223	faulty
37	685	303	faulty
38	698	197	faulty
39	699	311	faulty
40	712	257	faulty
41	722	193	faulty
42	735	259	faulty
43	738	314	faulty
44	753	113	faulty
45	767	286	faulty
46	771	264	faulty
47	780	137	faulty
48	784	131	faulty
49	798	132	faulty
50	820	152	faulty
51	834	157	faulty
52	858	163	faulty
53	888	91	faulty
54	891	156	faulty
55	911	79	faulty
56	939	99	faulty

John Kelleher and Brian Mac Namee and Aoife D'Arcy, Fundamentals of Machine Learning for Predictive Data Analytics

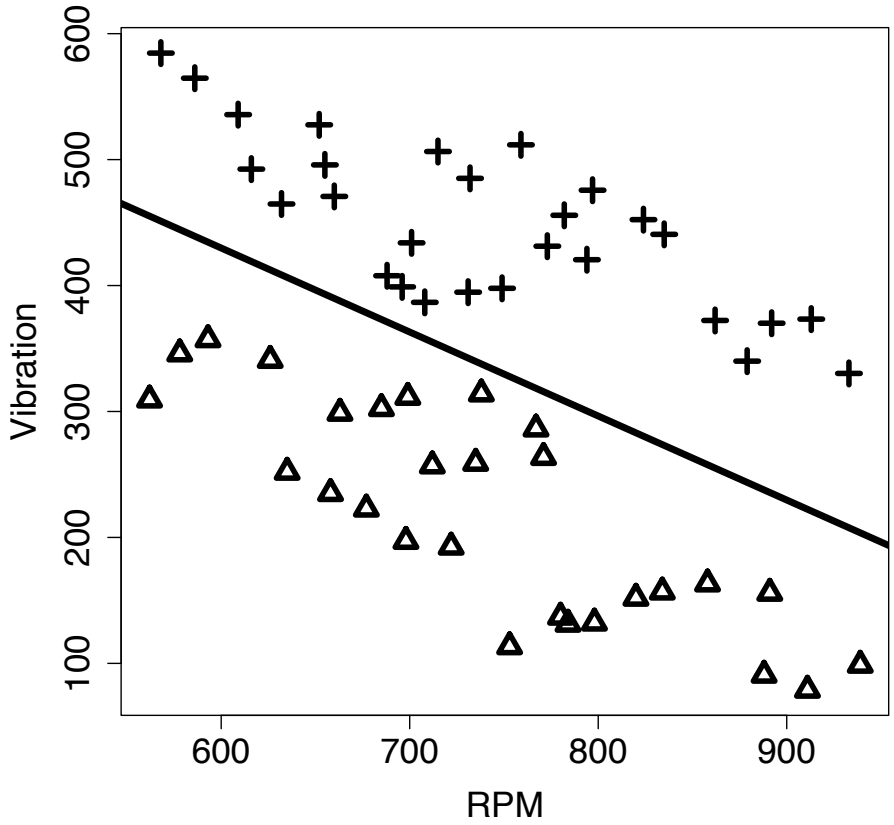
Classification - an example



John Kelleher and Brian Mac Namee and Aoife D'Arcy, Fundamentals of Machine Learning for Predictive Data Analytics

Content

Decision boundary



John Kelleher and Brian Mac Namee and Aoife D'Arcy, Fundamentals of Machine Learning for Predictive Data Analytics

Decision boundary

$$830 - 0.667 \times \text{RPM} - \text{Vibration} = 0$$

$$\theta_0 + \theta_1 x_1 + x_2 = 0$$

- For all "good" generators, we have: $\theta_0 + \theta_1 x_1 + x_2 >= 0$
- For all "faulty" generators, we have: $\theta_0 + \theta_1 x_1 + x_2 < 0$

Decision boundary

$$830 - 0.667 \times \text{RPM} - \text{Vibration} = 0$$

$$\theta_0 + \theta_1 x_1 + x_2 = 0$$

- For all "good" generators, we have: $\theta_0 + \theta_1 x_1 + x_2 >= 0$
- For all "faulty" generators, we have: $\theta_0 + \theta_1 x_1 + x_2 < 0$

Decision boundary

$$830 - 0.667 \times \text{RPM} - \text{Vibration} = 0$$

$$\theta_0 + \theta_1 x_1 + x_2 = 0$$

- For all "good" generators, we have: $\theta_0 + \theta_1 x_1 + x_2 >= 0$
- For all "faulty" generators, we have: $\theta_0 + \theta_1 x_1 + x_2 < 0$

Notation

$$\text{Let: } \theta = \begin{bmatrix} 1 \\ \theta_1 \\ \theta_0 \end{bmatrix}, x = \begin{bmatrix} x_2 \\ x_1 \\ 1 \end{bmatrix}$$

Then we have the decision boundary:

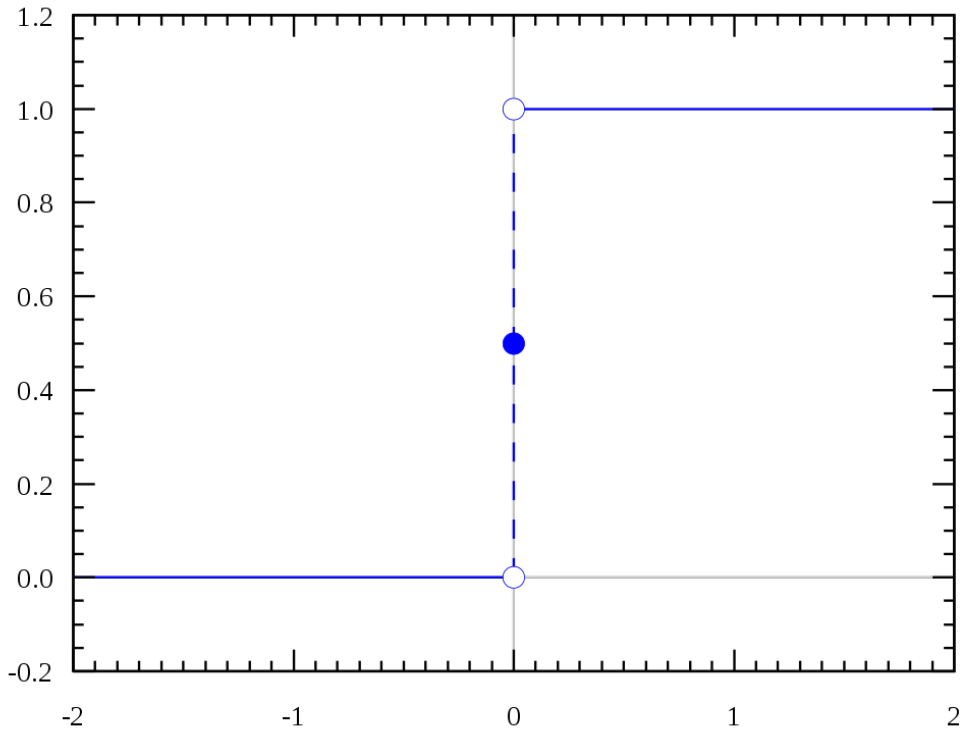
$$d_\theta(x) = \theta^T x = 0$$

Content

Heaviside step function

$$Heaviside(x) = \begin{cases} 1 & \text{if: } x >= 0 \\ 0 & \text{if: } x < 0 \end{cases}$$

Heaviside step function

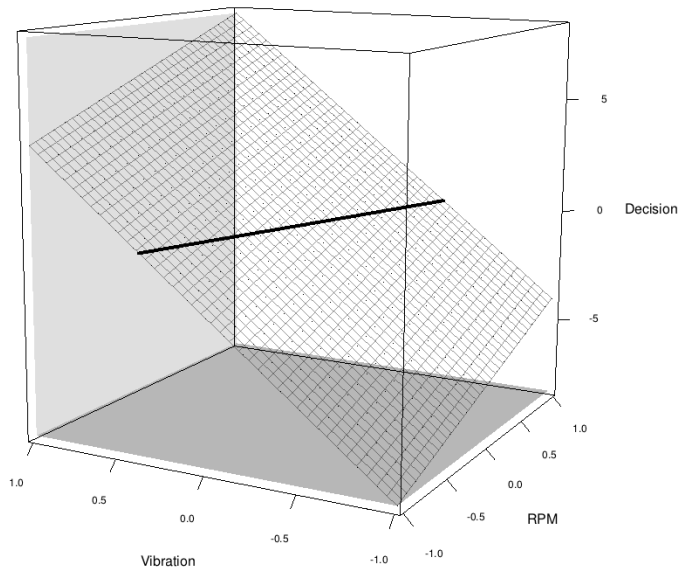


By Omegatron - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=801382

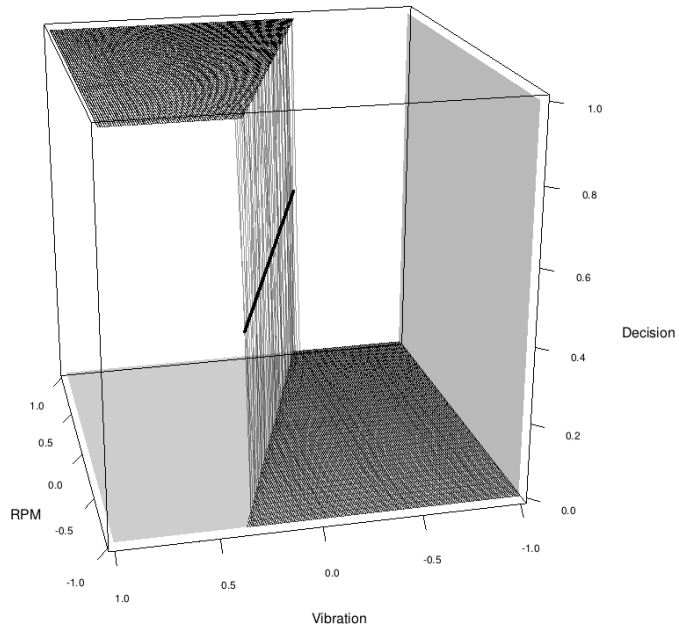
Model as a Heaviside step function

$$h_\theta(x) = Heaviside(d_\theta(x))$$
$$= \begin{cases} 1 & \text{if: } d_\theta(x) = \theta^T x >= 0; \text{ for } good \text{ generators} \\ 0 & \text{if: } d_\theta(x) = \theta^T x < 0; \text{ for } faulty \text{ generators} \end{cases}$$

Model as a Heaviside step function



(a)



(b)

$d_\theta(x)$

$h_\theta(x) = Heaviside(d_\theta(x))$

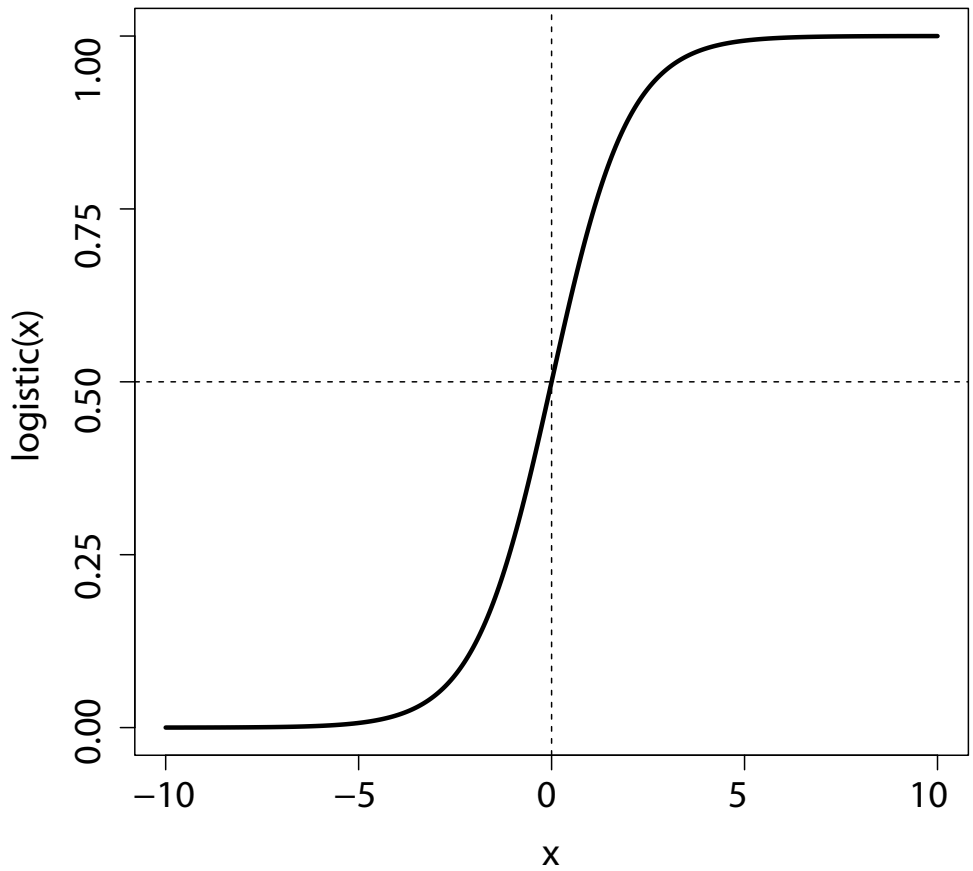
John Kelleher and Brian Mac Namee and Aoife D'Arcy, Fundamentals of Machine Learning for Predictive Data Analytics

Problem of the Heaviside step function

- Heaviside step function is not derivable and hard to optimize.
- An alternative is using a Logistic function (sigmoid function):

$$Logistic(x) = \frac{1}{1 + e^{-x}}$$

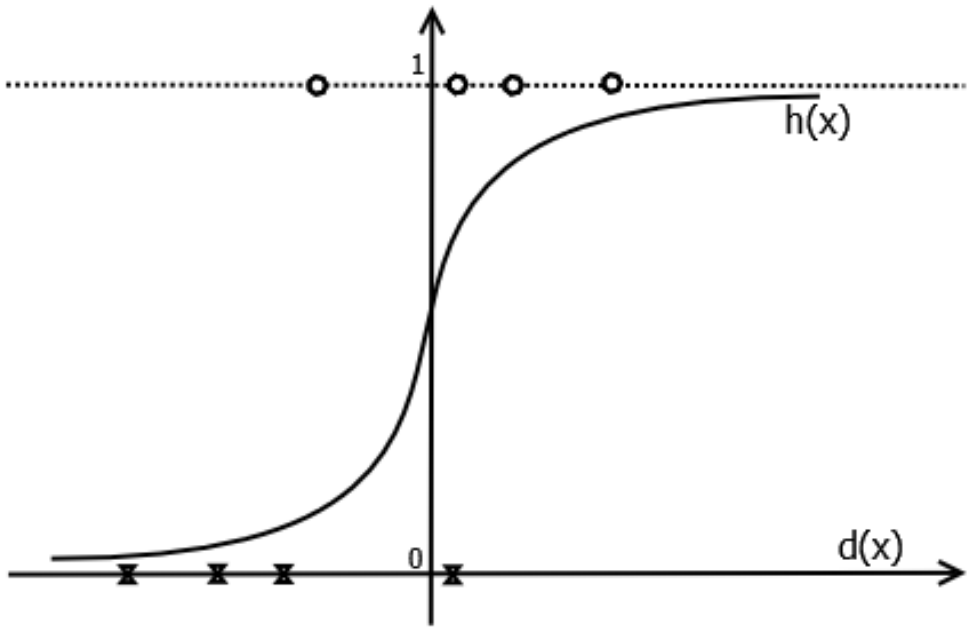
Logistic function



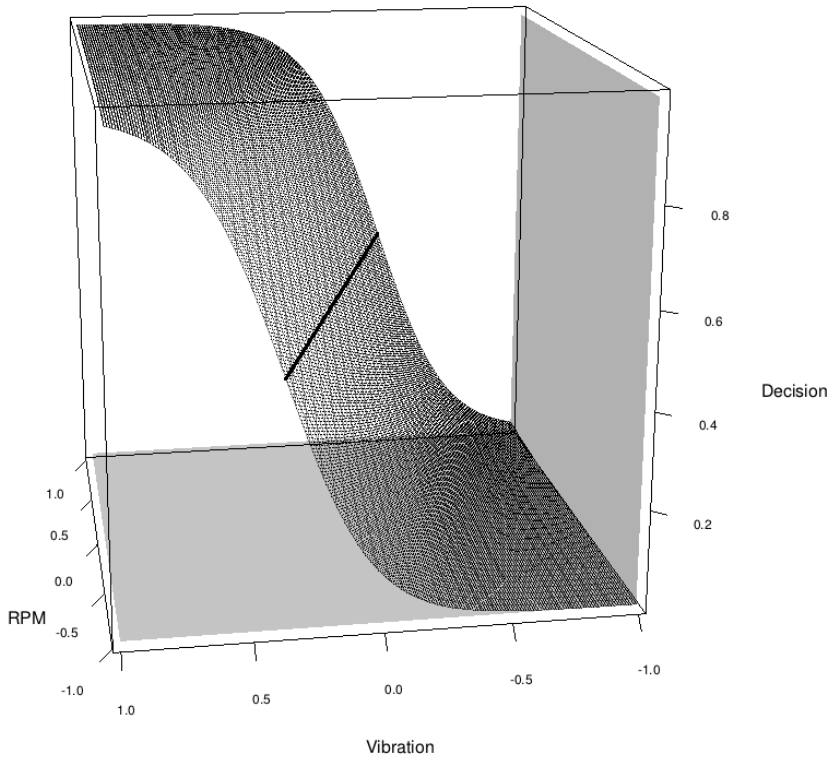
John Kelleher and Brian Mac Namee and Aoife D'Arcy, Fundamentals of Machine Learning for Predictive Data Analytics

Model as a logistic function

$$h_\theta(x) = Logistic(d_\theta(x)) = \frac{1}{1 + e^{-d_\theta(x)}} = \frac{1}{1 + e^{-\theta^T x}}$$



Model as a logistic function



John Kelleher and Brian Mac Namee and Aoife D'Arcy, Fundamentals of Machine Learning for Predictive Data Analytics

Content

Cost function

- The objective of a learning algorithm is to search a model to best predict the target feature on the training data given the inductive bias.
- One way to achieve this goal is to minimize the errors of the prediction over the training data.
- A cost function (loss function) is defined as the sum of the errors over the training samples.

Cost function

- A possible cost function:

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

- It is not a good cost function for Logistic regression because it is non-convex for a Logistic function.

Convex vs. non convex functions

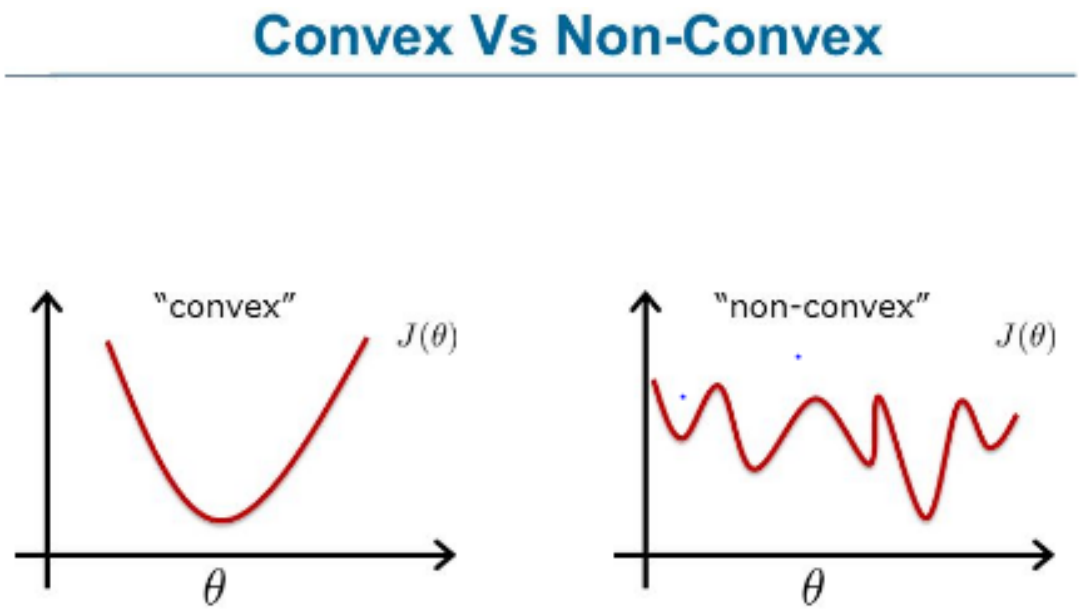


figure source: https://www.fromthegenesis.com/artificial-neural-network-part-7/

Cost function for Logisitc regression

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} Cost(h_\theta(x^{\{i\}}), y^{\{i\}})$$

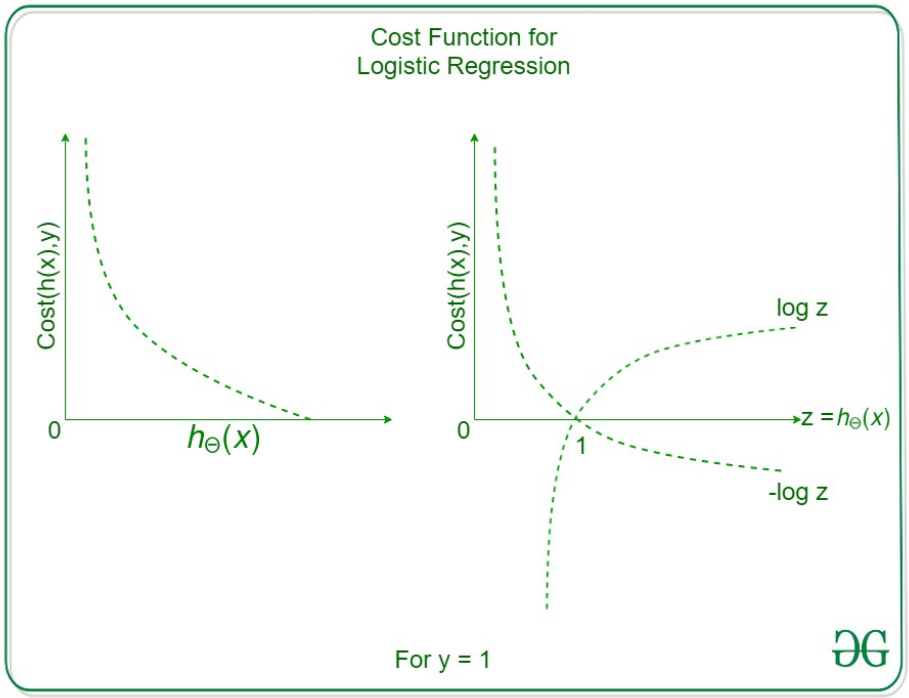
where:

$$Cost(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$
$$= -\left[y \log(h_\theta(x)) + (1 - y) \log(1 - h_\theta(x))\right]$$

$y^{\{*\}} \in \{0, 1\}$

Cost function for Logisitc regression

$$Cost(h_\theta(x), y) = -\left[y \log(h_\theta(x)) + (1 - y) \log(1 - h_\theta(x))\right]$$



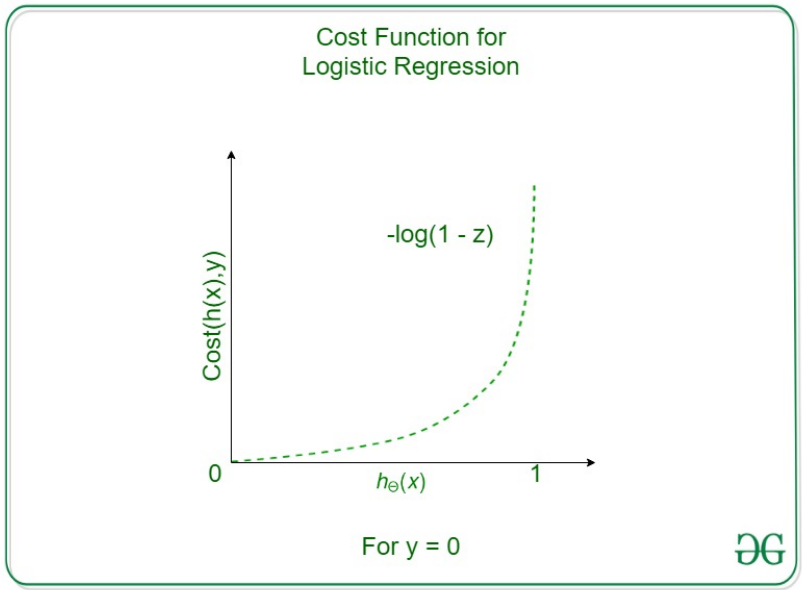


figure source: https://www.geeksforgeeks.org/ml-cost-function-in-logistic-regression/

Content

Gradient descend

- Now we have a cost function which defines the errors of a model over the training data.
- Next we need to search all the possible models to find a model with the minimal cost.
- We use a gradient descend algorithm for this purpose.

Gradient descend

Have some function $J(\theta_0, \theta_1)$

Want $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$

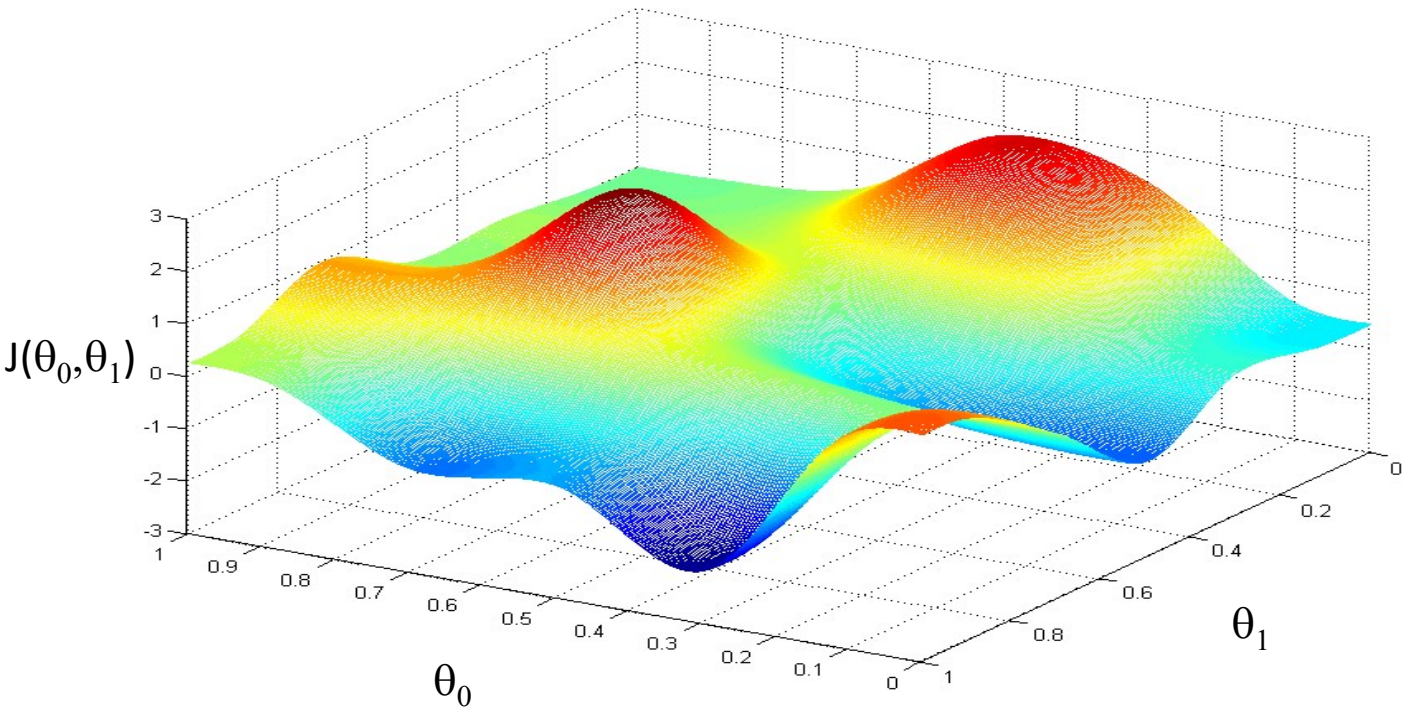
Outline:

- Start with some θ_0, θ_1
- Keep changing θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$

  until we hopefully end up at a minimum

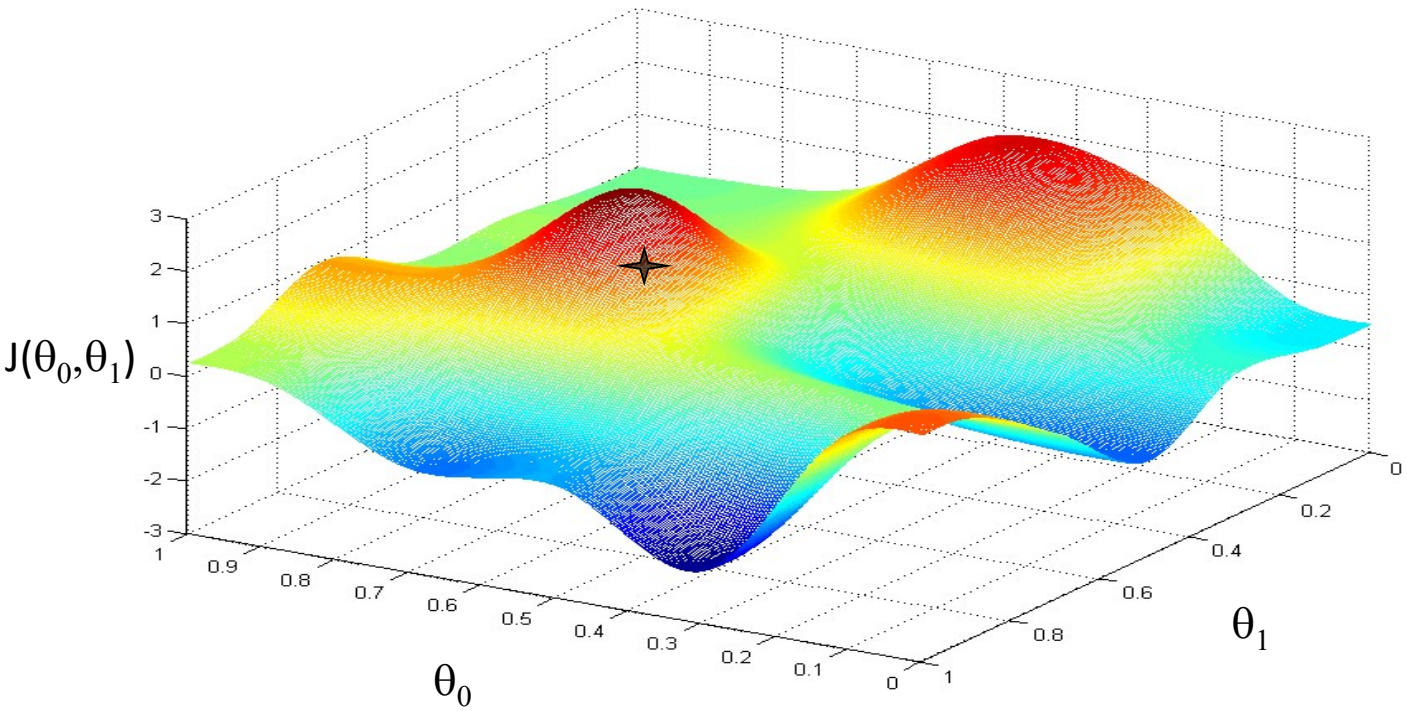
Andrew Ng, Machine Learning, Coursera course

Gradient descend



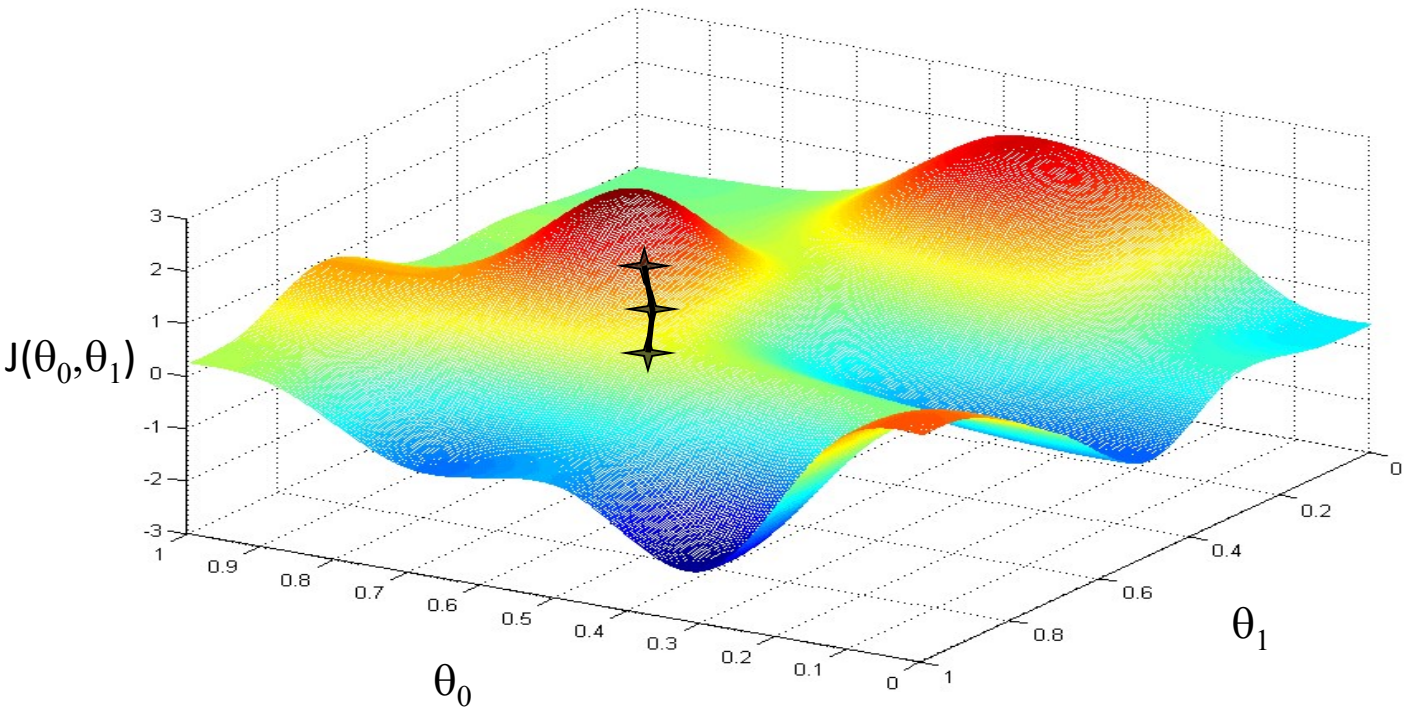
Andrew Ng, Machine Learning, Coursera course

Gradient descend



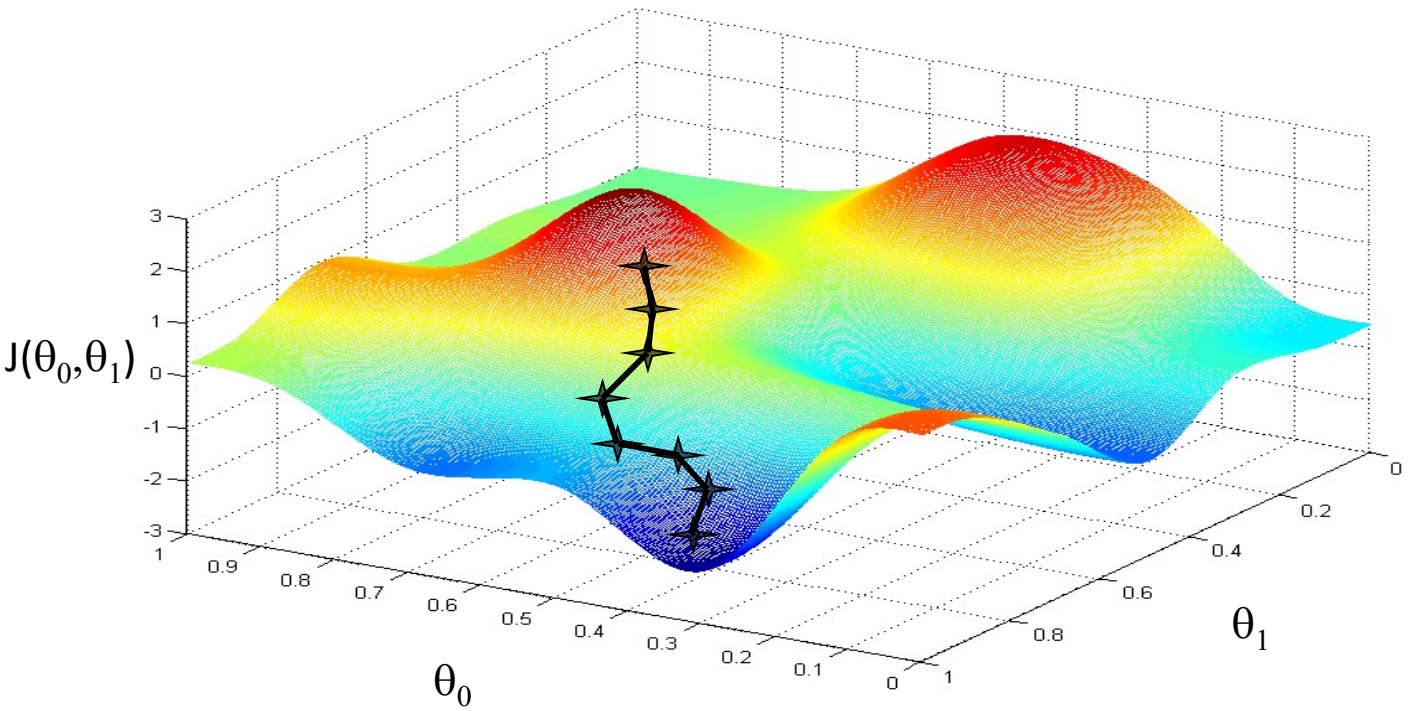
Andrew Ng, Machine Learning, Coursera course

Gradient descend



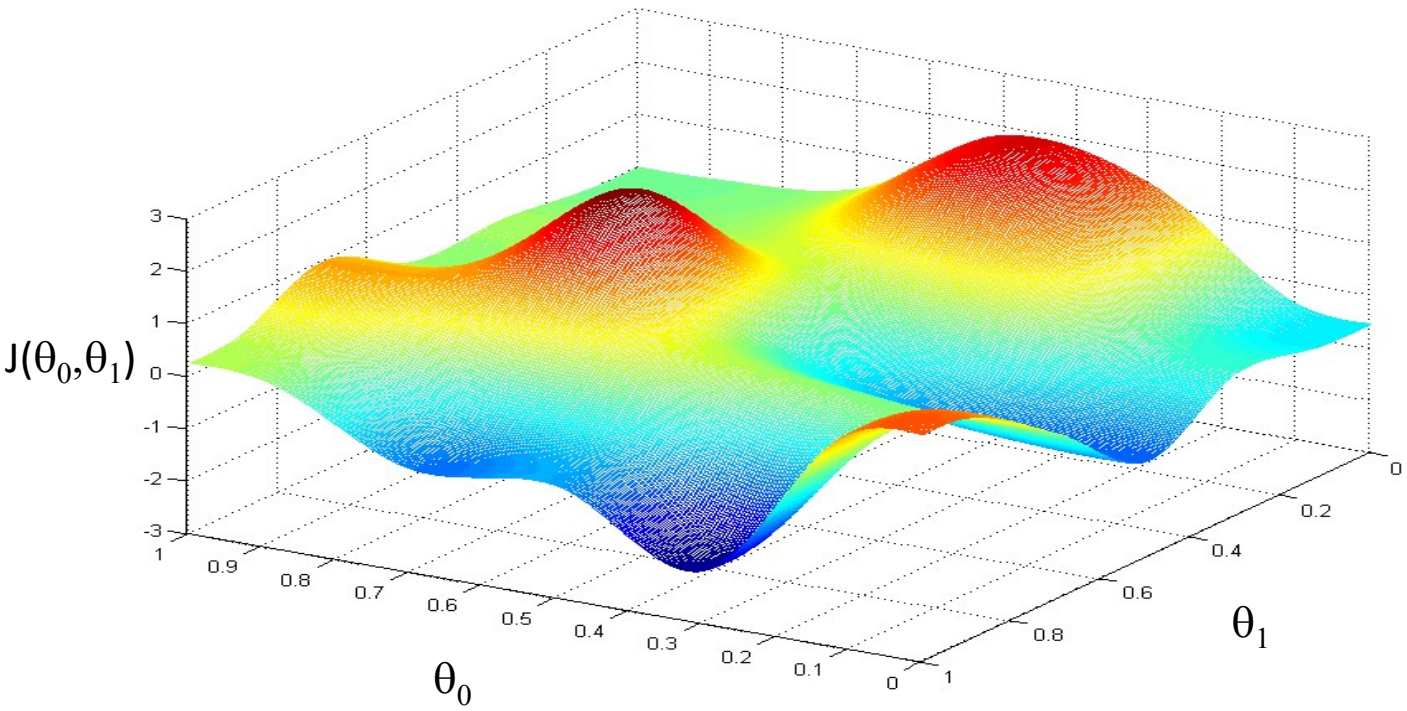
Andrew Ng, Machine Learning, Coursera course

Gradient descend



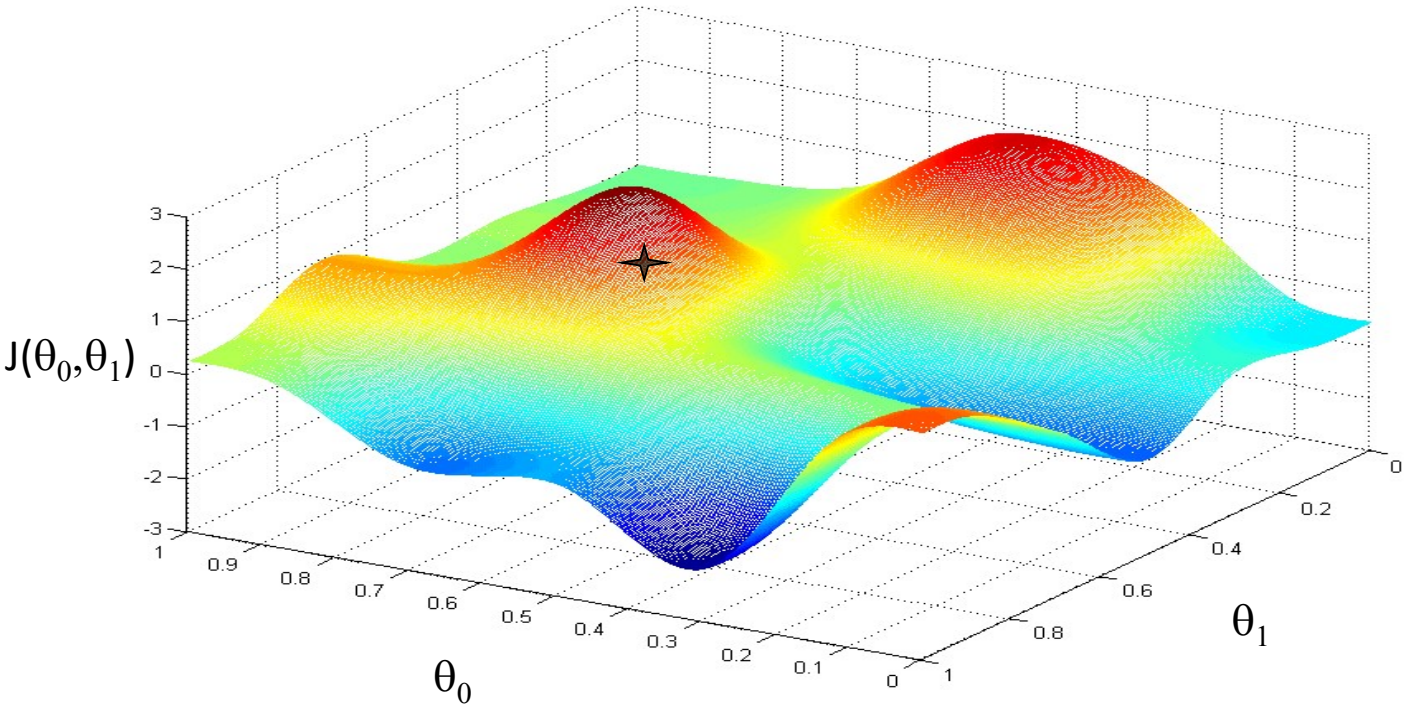
Andrew Ng, Machine Learning, Coursera course

Gradient descend



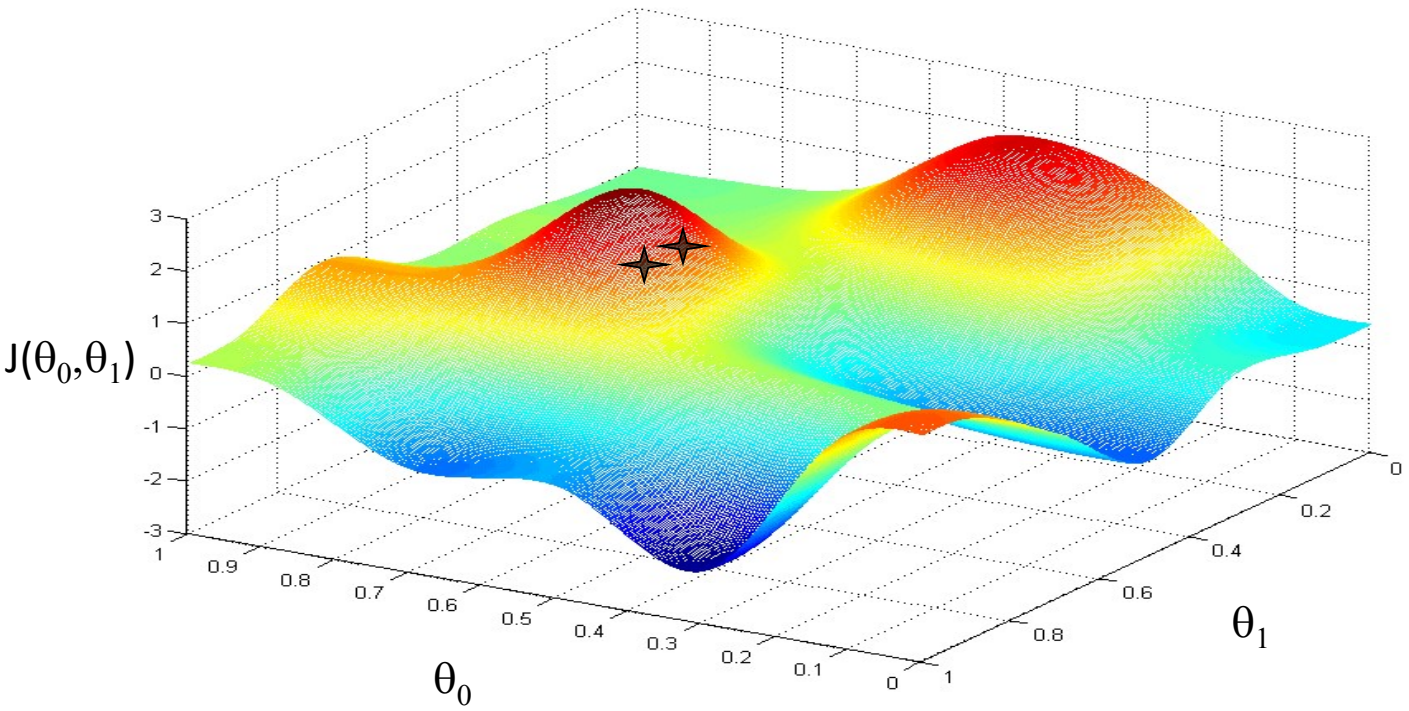
Andrew Ng, Machine Learning, Coursera course

Gradient descend



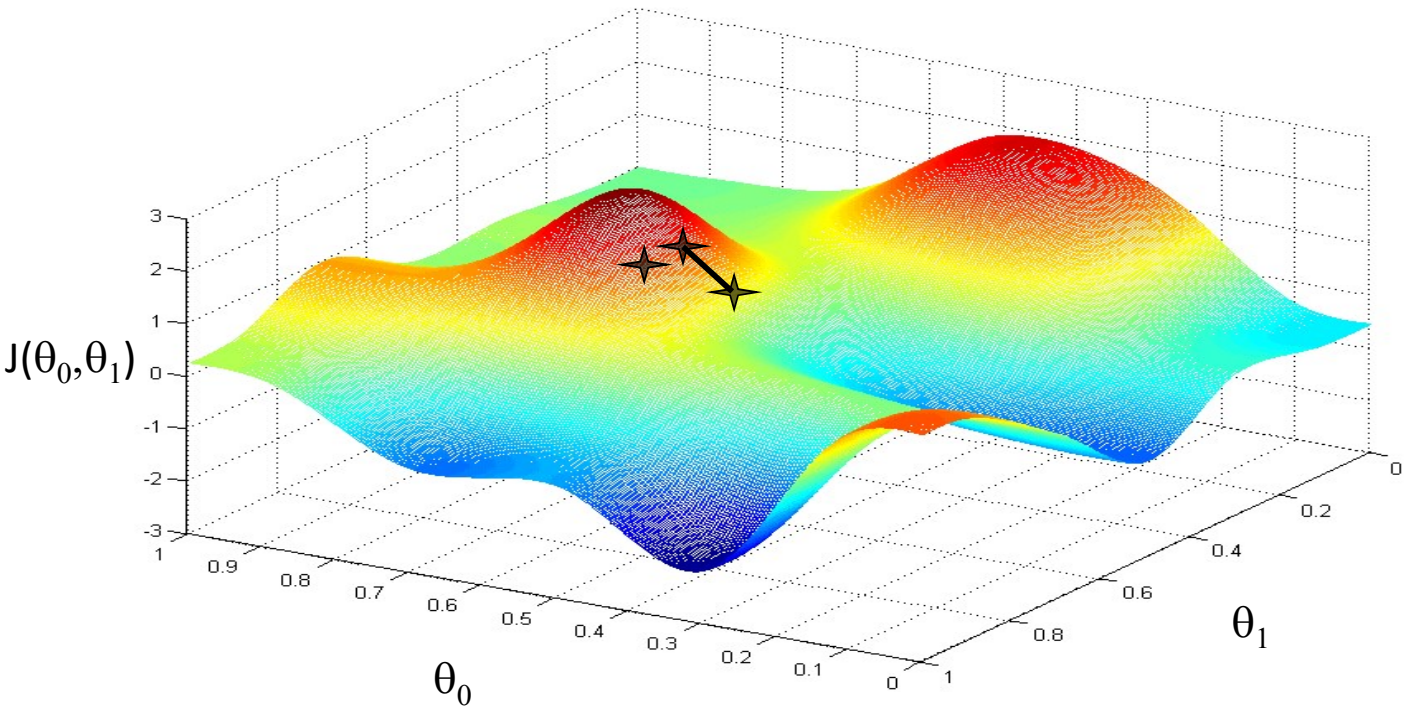
Andrew Ng, Machine Learning, Coursera course

Gradient descend



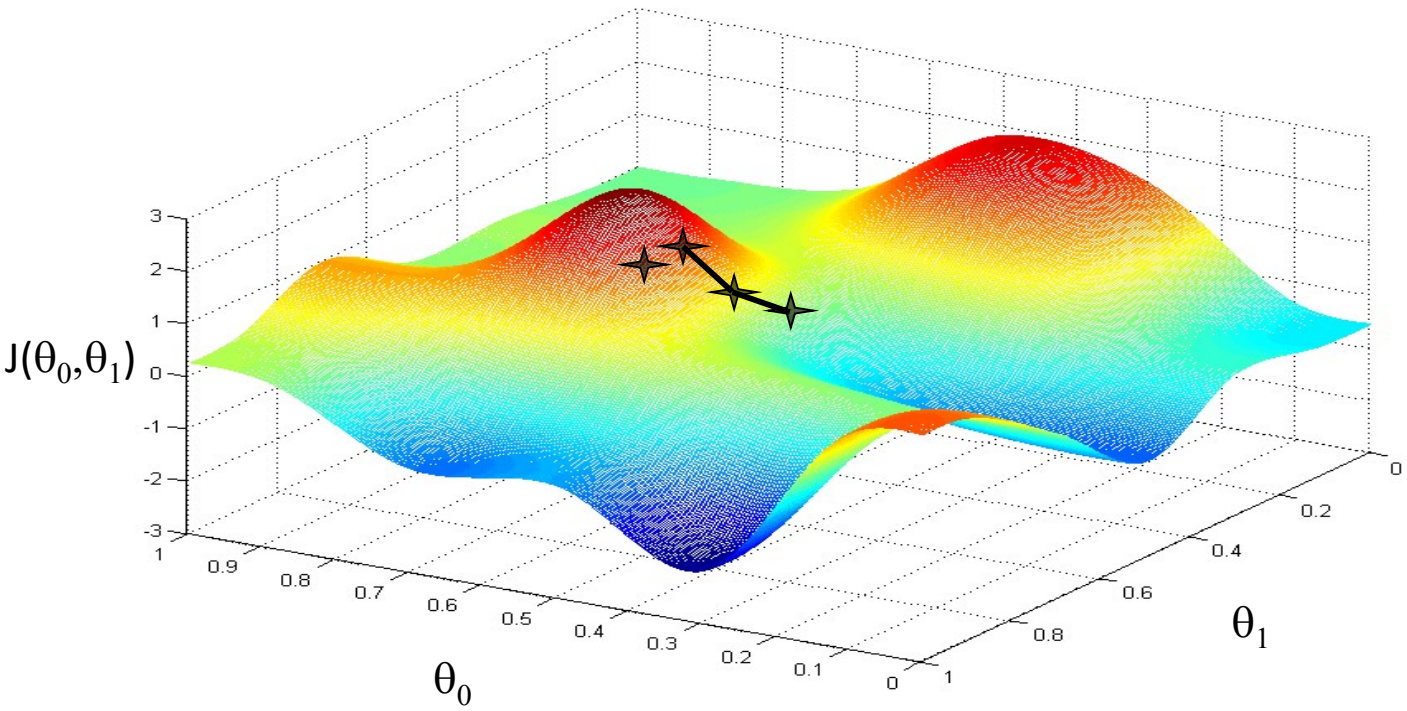
Andrew Ng, Machine Learning, Coursera course

Gradient descend



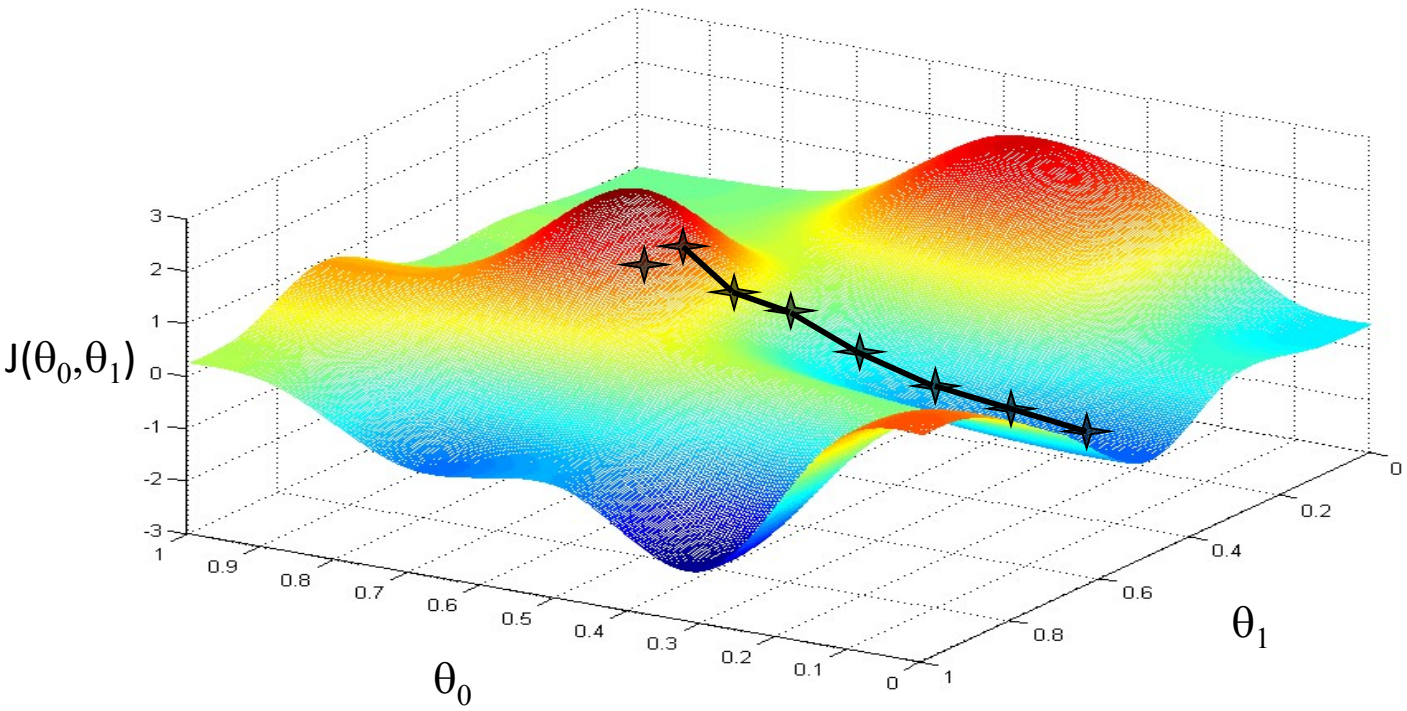
Andrew Ng, Machine Learning, Coursera course

Gradient descend



Andrew Ng, Machine Learning, Coursera course

Gradient descend



Andrew Ng, Machine Learning, Coursera course

Gradient descend algorithm

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad \begin{array}{l}(\text{simultaneously update} \\ j = 0 \text{ and } j = 1)\end{array}$$

}

Andrew Ng, Machine Learning, Coursera course

Gradient descend algorithm

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \qquad (\text{for } j = 0 \text{ and } j = 1)$$

}

Correct: Simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

Andrew Ng, Machine Learning, Coursera course

Gradient descend algorithm

Gradient descent algorithm

repeat until convergence {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad (\text{for } j = 0 \text{ and } j = 1)$$

}

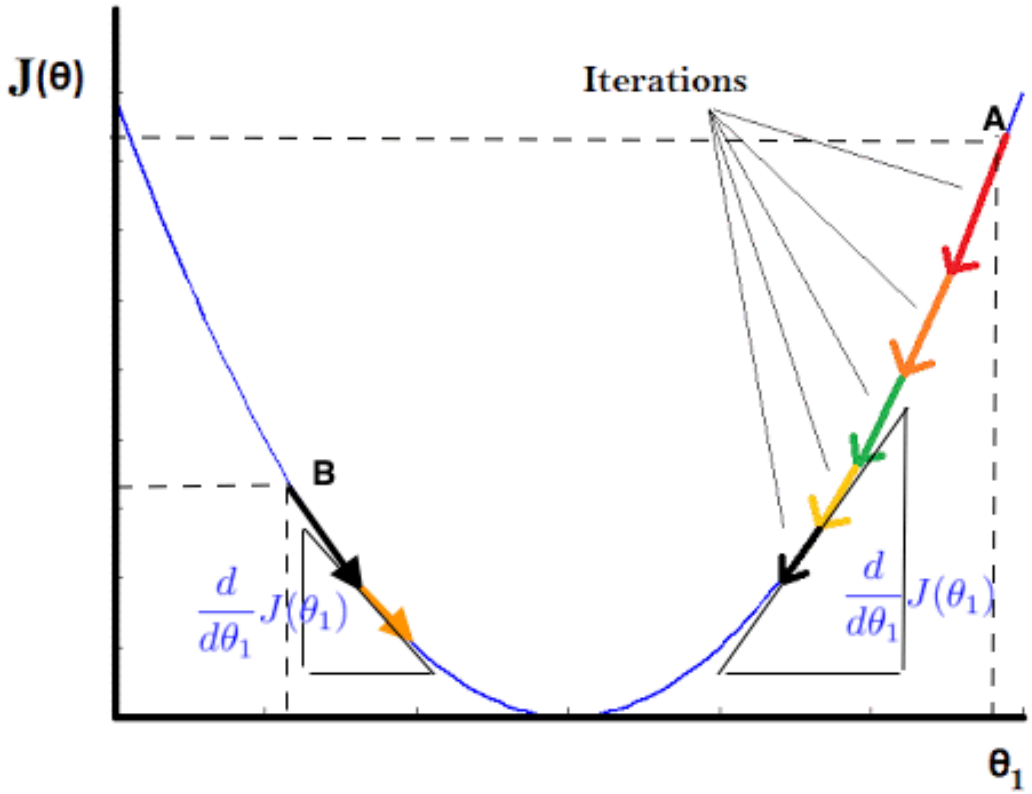
Correct: Simultaneous update

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

Incorrect:

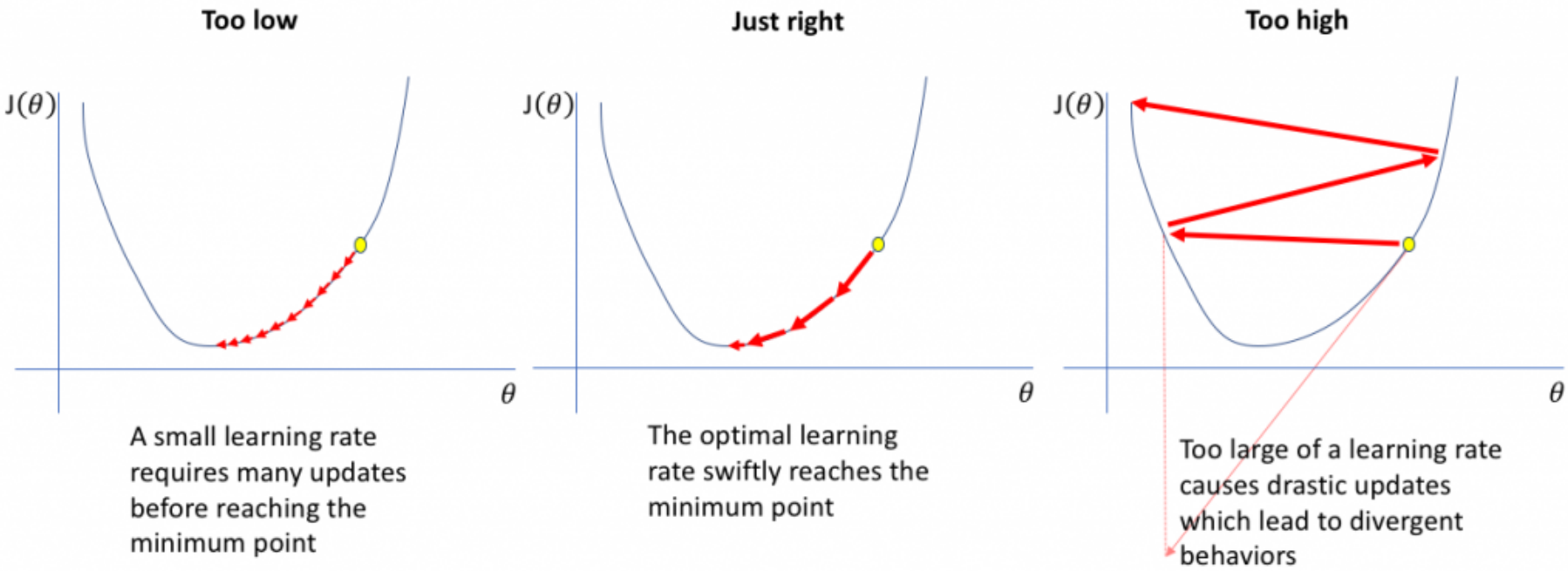
$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_1 := \text{temp1}$$

Andrew Ng, Machine Learning, Coursera course



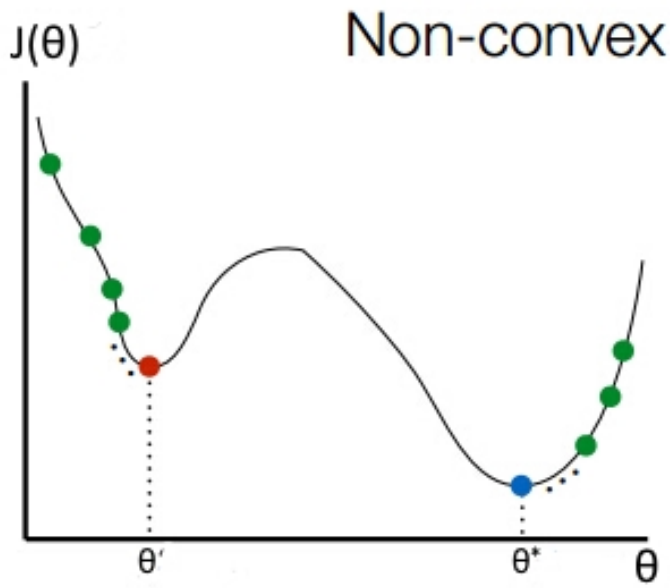
- The derivation $\frac{\partial J(\theta)}{\partial \theta}$ gives the direction of the movement.
- The learning rate α is used to adjust the size for each step.

figure source: https://machinelearningmedium.com/2017/08/15/gradient-descent/



The influence of the learning rate.

figure source: https://ithelp.ithome.com.tw/m/articles/10204032



A non-convex error surface may lead to local optima.

Gradient descend for Logistic regression

Model

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

Parameters

$$\theta_0, \theta_1$$

Cost Function

$$J(\theta) = -\frac{1}{n} \sum_{i=1}^{n} \left[y^{\{i\}} \log(h_\theta(x^{\{i\}})) + (1 - y^{\{i\}}) \log(1 - h_\theta(x^{\{i\}})) \right]$$

Goal

$$\underset{\theta_0, \theta_1}{\text{minimize}}\ J(\theta)$$

Gradient descend for Logistic regression

Gradient descent:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{\{i\}}\log(h_\theta(x^{\{i\}})) + (1-y^{\{i\}})\log(1-h_\theta(x^{\{i\}}))\right]$$

Want $\min_\theta J(\theta)$:

Repeat {

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

(simultaneously update all θ_j)

}

Gradient descend for Logistic regression

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{\{i\}}\log(h_\theta(x^{\{i\}})) + (1-y^{\{i\}})\log(1-h_\theta(x^{\{i\}}))\right]$$

$$\begin{aligned}\frac{\partial}{\partial\theta_j}J(\theta) &= -\frac{1}{n}\frac{\partial}{\partial\theta_j}\sum_{i=1}^{n}\left[y^{\{i\}}\log(h_\theta(x^{\{i\}})) + (1-y^{\{i\}})\log(1-h_\theta(x^{\{i\}}))\right] \\ &= \cdots\cdots \\ &= \frac{1}{n}\sum_{i=1}^{n}(h_\theta(x^{\{i\}}) - y^{\{i\}})x_j^{\{i\}}\end{aligned}$$

Gradient descend for Logistic regression

Gradient descent:

$$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\left[y^{\{i\}}\log(h_\theta(x^{\{i\}})) + (1-y^{\{i\}})\log(1-h_\theta(x^{\{i\}}))\right]$$

Want $\min_\theta J(\theta)$:

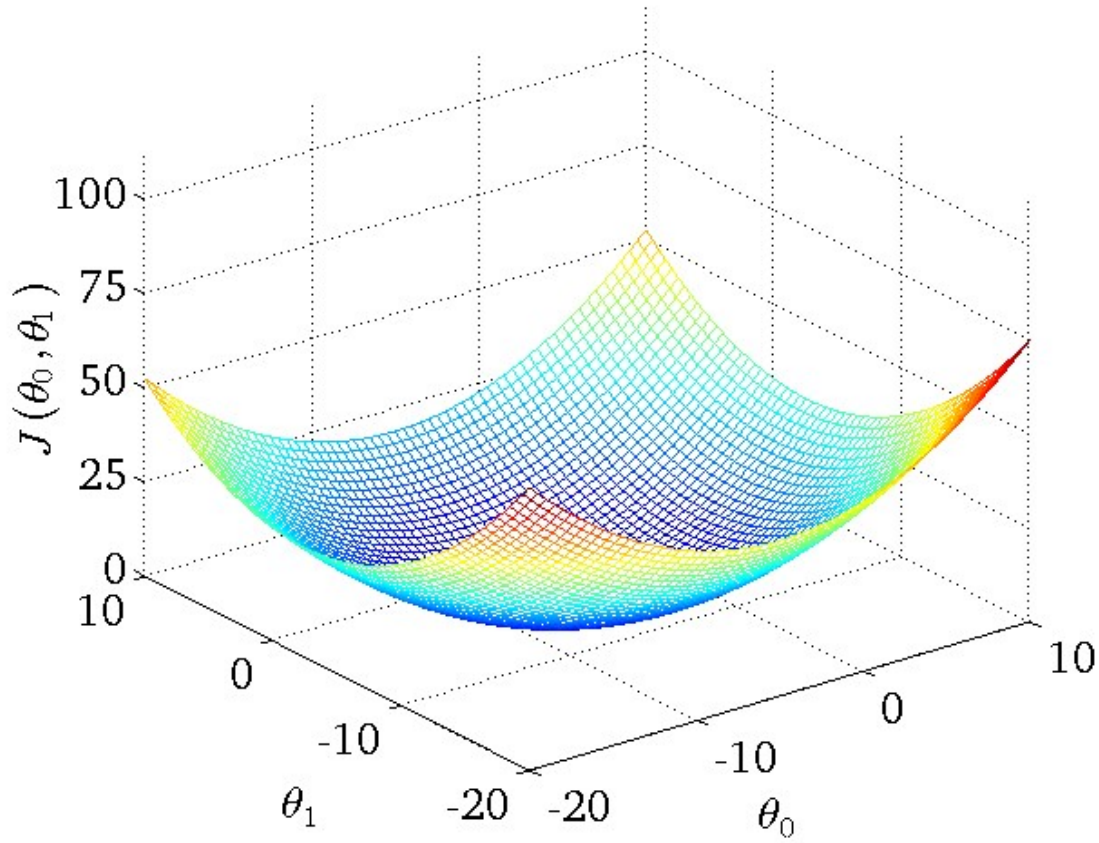
Repeat {

$\theta_j = \theta_j - \alpha\frac{1}{n}\sum_{i=1}^{n}(h_\theta(x^{\{i\}}) - y^{\{i\}})x_j^{\{i\}}$

(simultaneously update all θ_j)

}

Gradient descend for logistic regression



Fortunately the error surface for logistic regression is convex.

Andrew Ng, Machine Learning, Coursera course