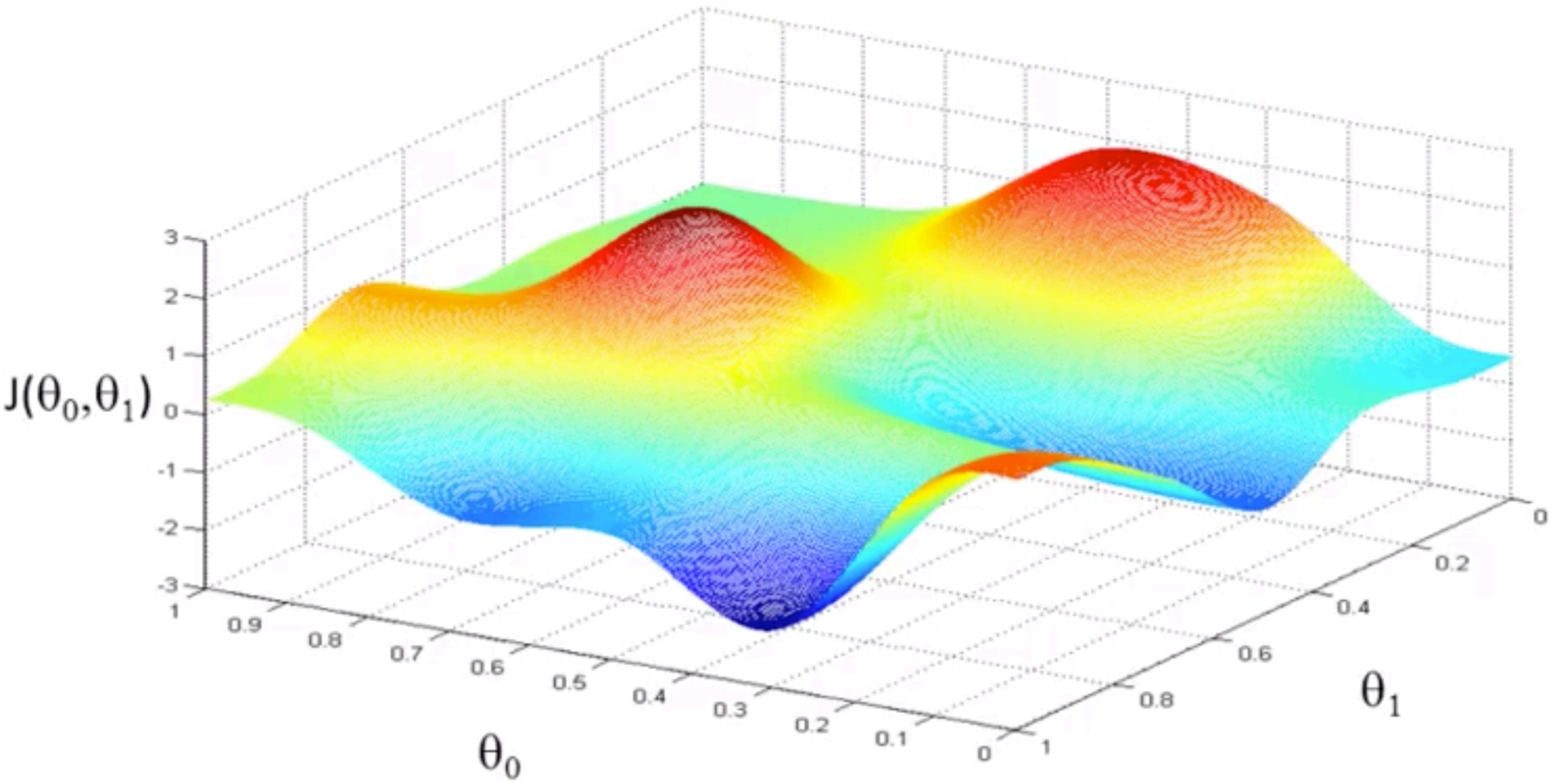
Iterative Methods

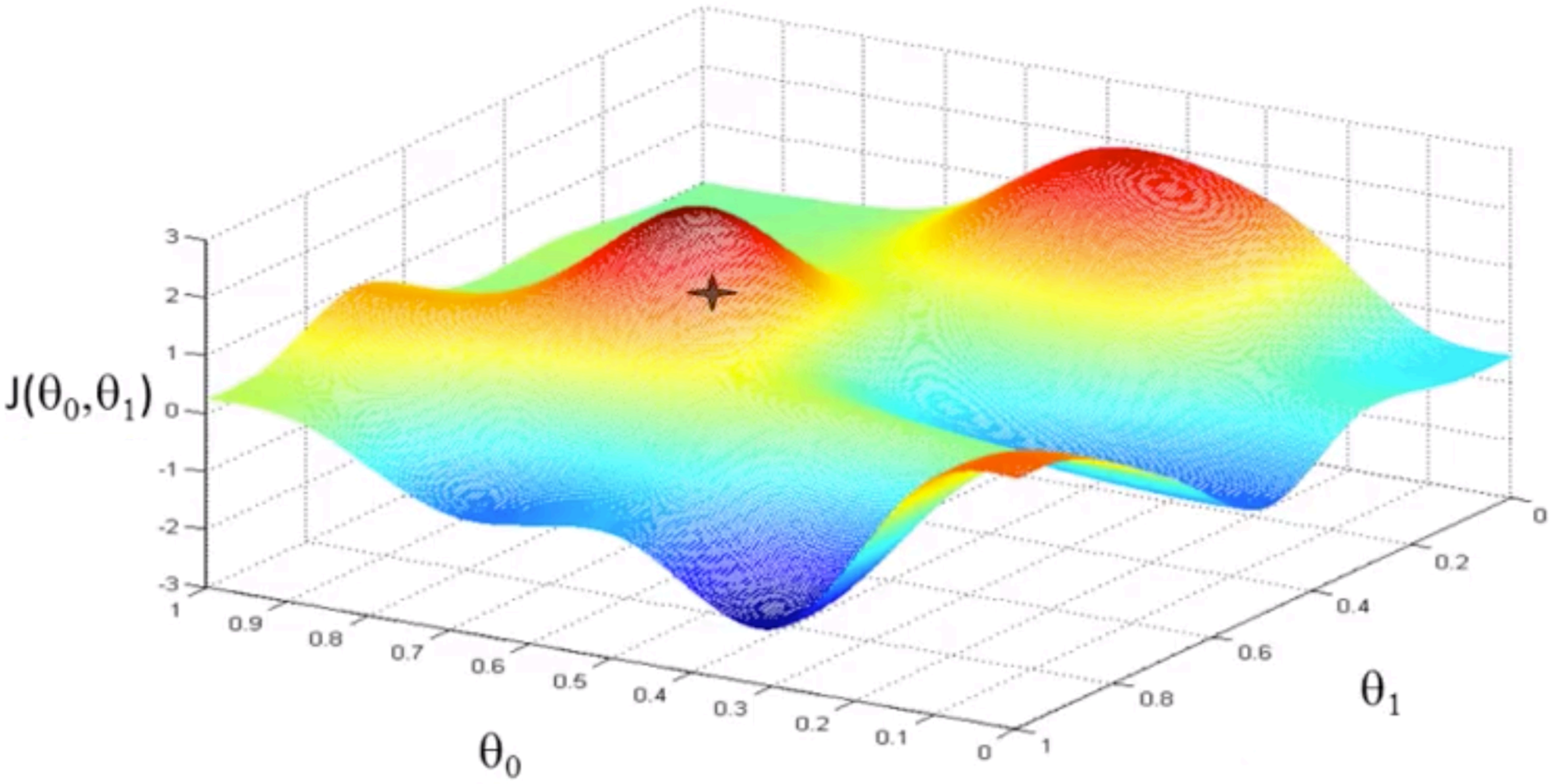
Teeradaj Racharak (เอ็กซ์)
r.teeradaj@gmail.com

Gradient Descent (Reviewed)



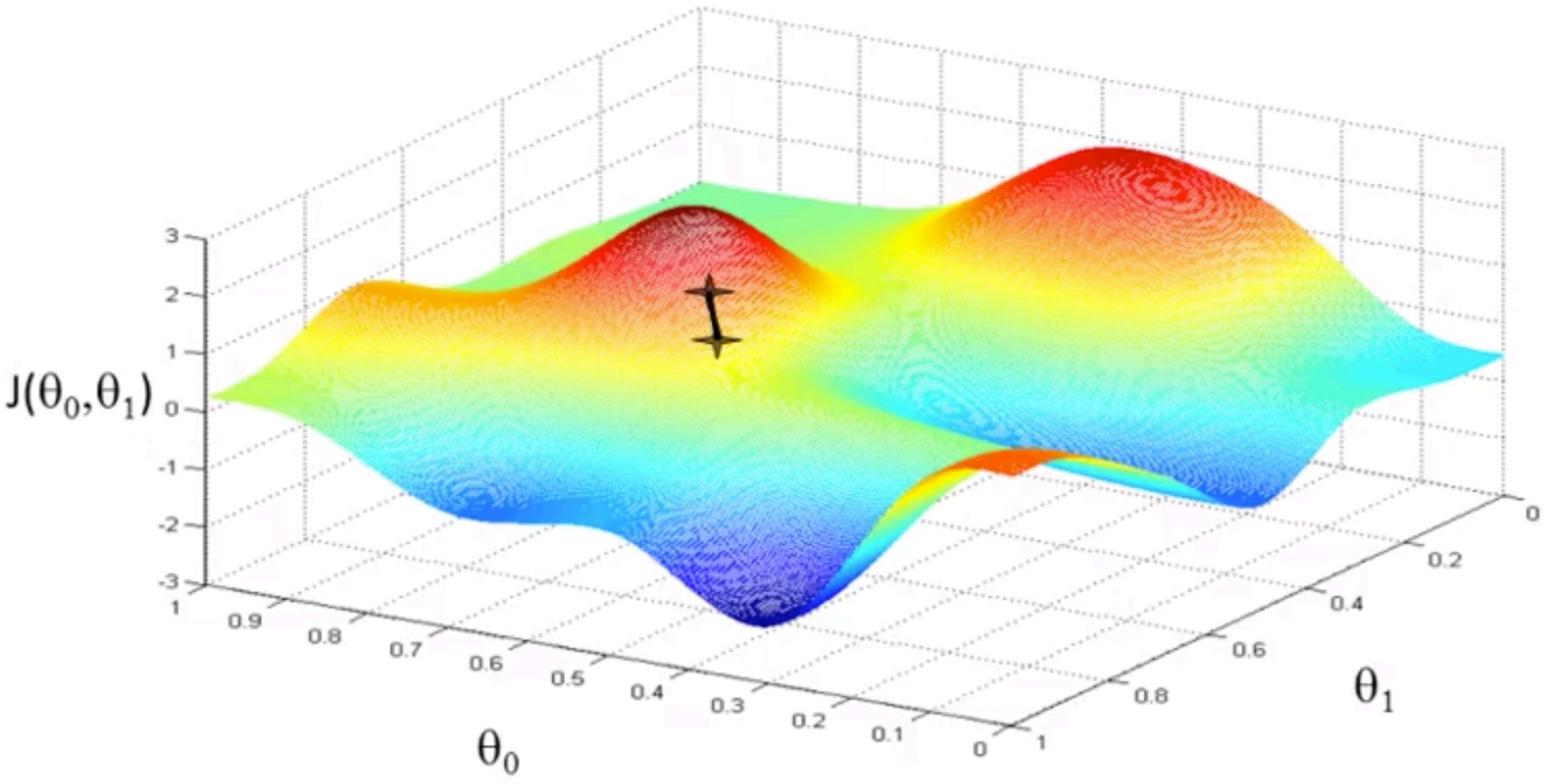
(source: Andrew Ng, machine learning)

Gradient Descent (Reviewed)



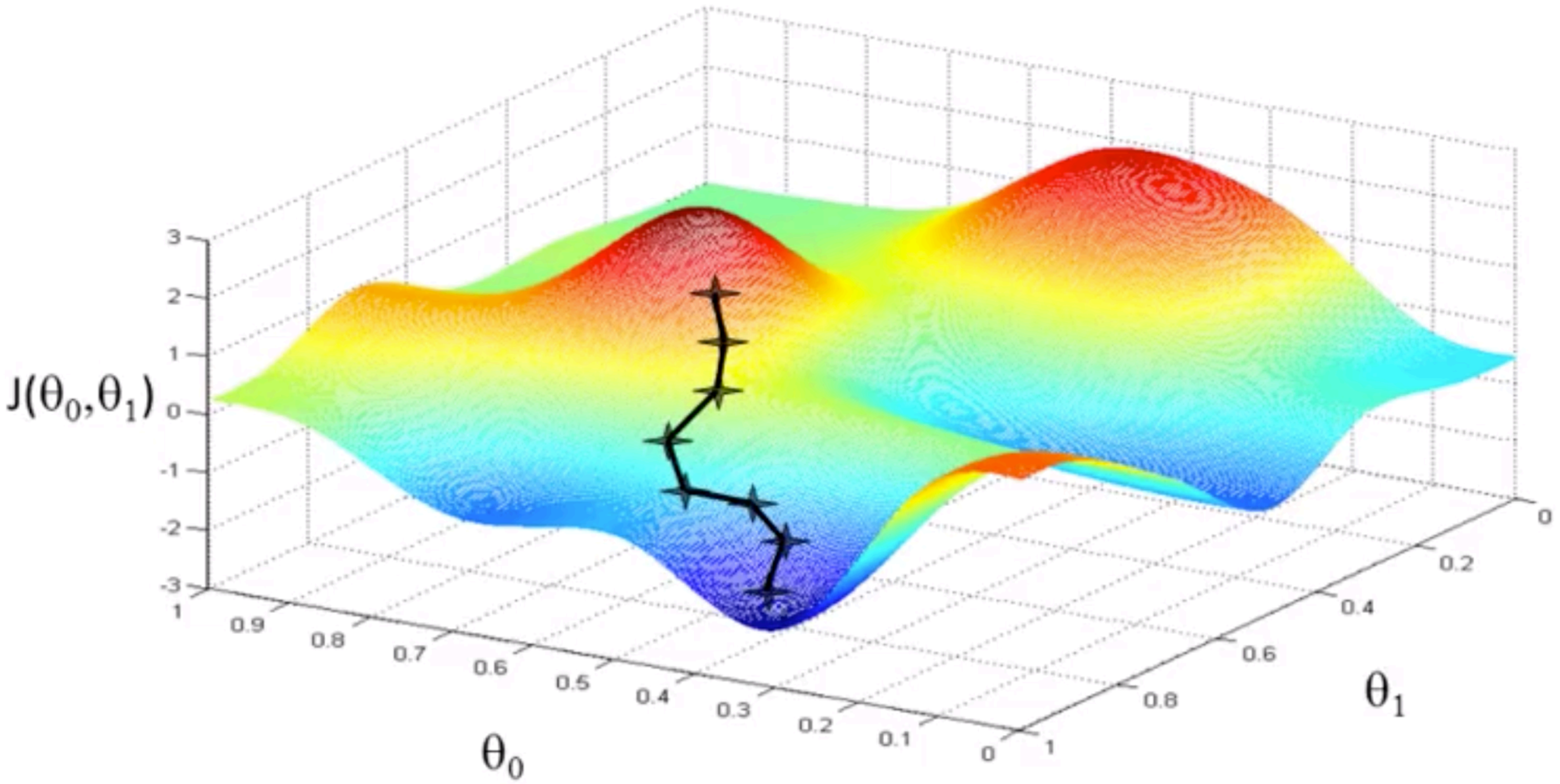
(source: Andrew Ng, machine learning)

Gradient Descent (Reviewed)



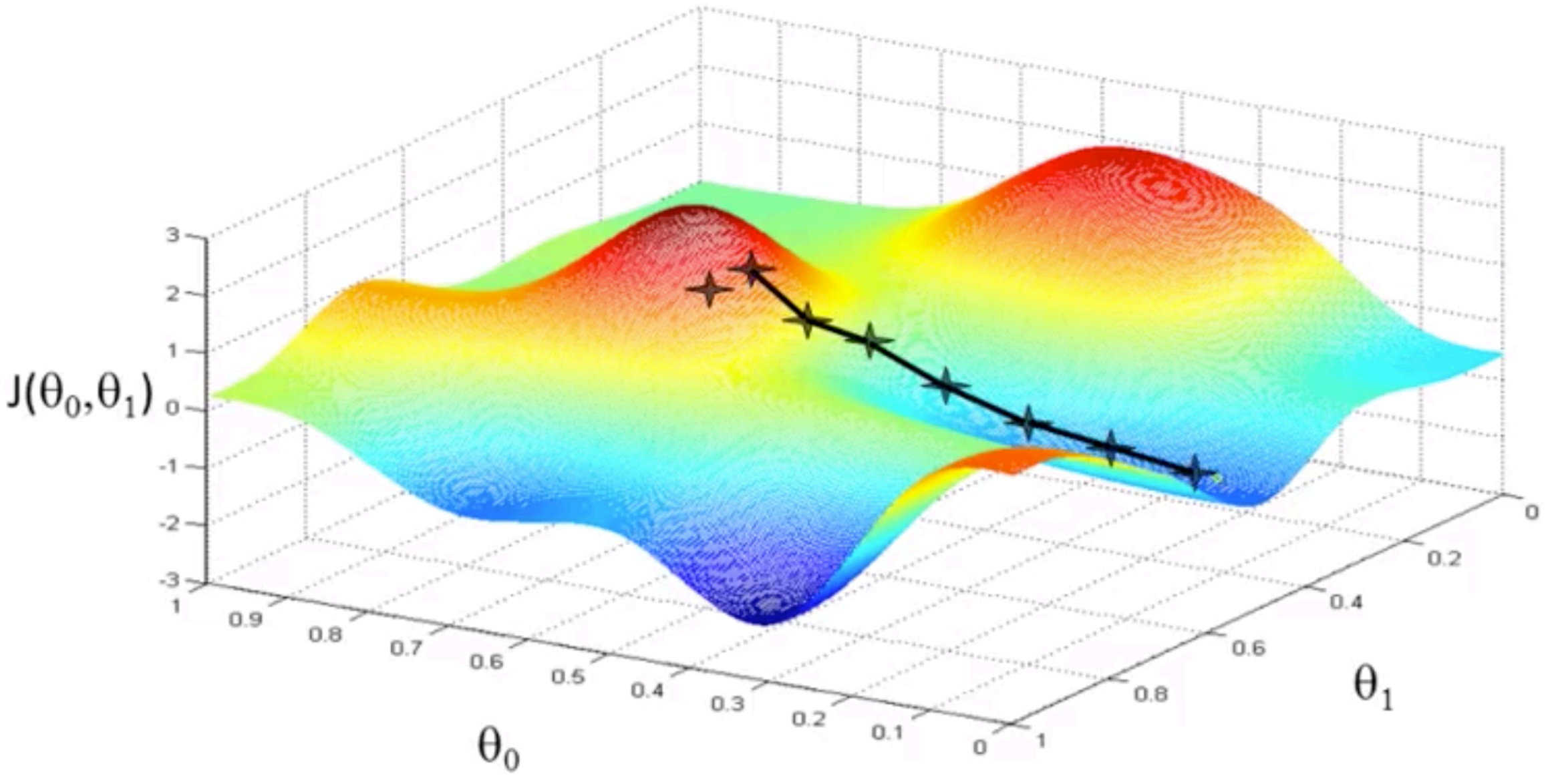
(source: Andrew Ng, machine learning)

Gradient Descent (Reviewed)



(source: Andrew Ng, machine learning)

Gradient Descent (Reviewed)



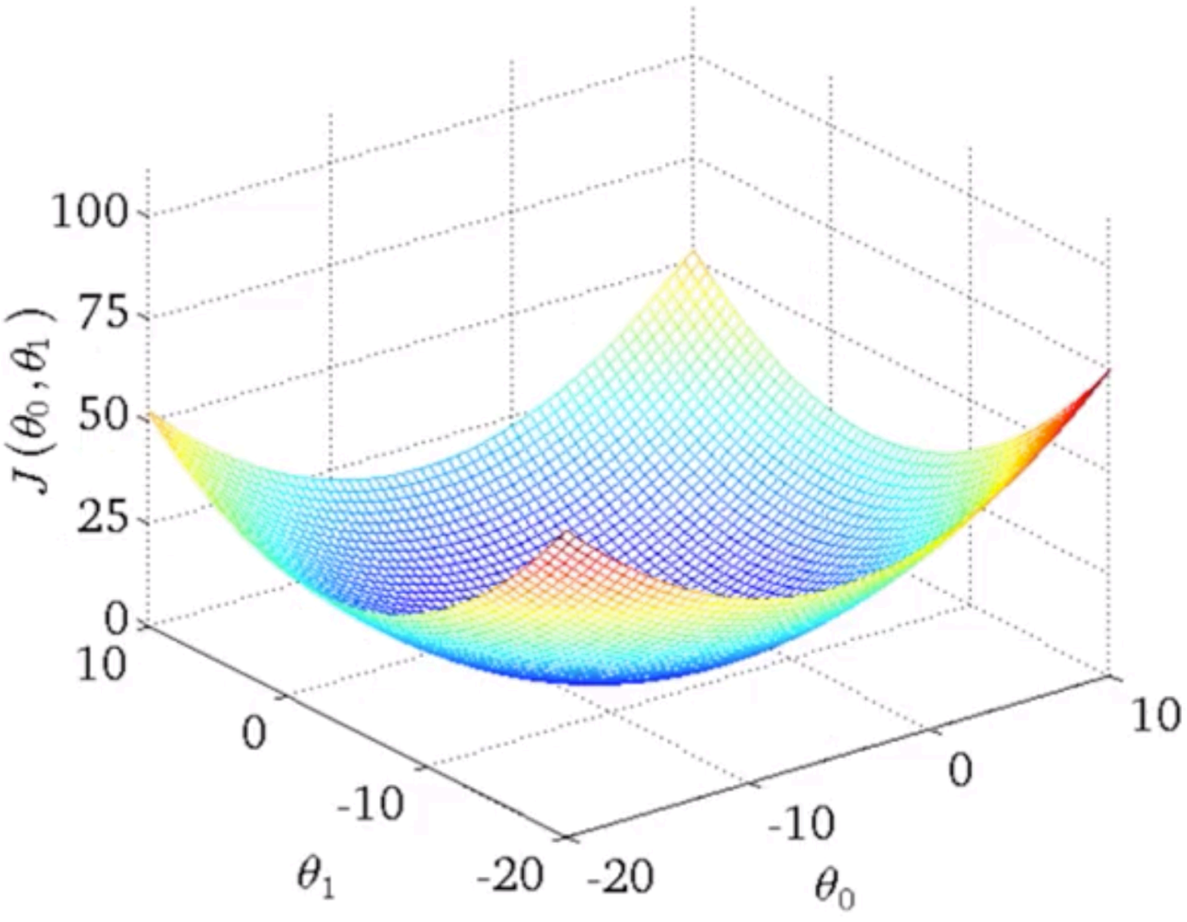
Gradient descent has an ‘interesting’ property !

(source: Andrew Ng, machine learning)

Linear Regression

The cost function of linear regression has always the bowl shape like this.

i.e. ‘convex function’



GD Algorithm (Reviewed)

repeat until convergence {

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

}

$$\because \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) = \frac{\partial}{\partial \theta_j} [\frac{1}{2m} \Sigma_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2]$$

$$= \frac{\partial}{\partial \theta_j} [\frac{1}{2m} \Sigma_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2]$$

GD Algorithm (Reviewed)

For our cost function, think of it this way:

$$g(\theta_0, \theta_1) = \frac{1}{2m}\Sigma_{i=1}^{m}(f(\theta_0, \theta_1)^{(i)})^2$$

$$f(\theta_0, \theta_1)^{(i)} = \theta_0 + \theta_1 x^{(i)} - y^{(i)}$$

$$\therefore g(f(\theta_0, \theta_1)^{(i)}) = \frac{1}{2m}\Sigma_{i=1}^{m}(\theta_0 + \theta_1 x^{(i)} - y^{(i)})^2$$

GD Algorithm (Reviewed)

Then, the partial derivatives work like this:

$$\frac{\partial}{\partial \theta_0} g(f(\theta_0, \theta_1)^{(i)}) = \frac{\partial}{\theta_0} g(\theta_0, \theta_1) \cdot \frac{\partial}{\partial \theta_0} f(\theta_0, \theta_1)^{(i)} \qquad \text{(by the chain rule)}$$

GD Algorithm (Reviewed)

Then, the partial derivatives work like this:

$$\frac{\partial}{\partial\theta_0} g(f(\theta_0,\theta_1)^{(i)}) = \frac{\partial}{\theta_0} g(\theta_0,\theta_1) \; \frac{\partial}{\partial\theta_0} f(\theta_0,\theta_1)^{(i)} \qquad \text{(by the chain rule)}$$

$$\frac{\partial}{\partial\theta_0} g(\theta_0,\theta_1) = \frac{\partial}{\partial\theta_0} \frac{1}{2m} \Sigma_{i=1}^{m} (f(\theta_0,\theta_1)^{(i)})^2$$

$$= 2 \times \frac{\partial}{\partial\theta_0} \frac{1}{2m} \Sigma_{i=1}^{m} (f(\theta_0,\theta_1)^{(i)})^{2-1} \quad = \frac{1}{m} \Sigma_{i=1}^{m} f(\theta_0,\theta_1)^{(i)}$$

GD Algorithm (Reviewed)

Then, the partial derivatives work like this:

$$\frac{\partial}{\partial \theta_0} g(f(\theta_0, \theta_1)^{(i)}) = \frac{\partial}{\partial \theta_0} g(\theta_0, \theta_1) \cdot \frac{\partial}{\partial \theta_0} f(\theta_0, \theta_1)^{(i)} \quad \text{(by the chain rule)}$$

$$\frac{\partial}{\partial \theta_0} f(\theta_0, \theta_1)^{(i)} = \frac{\partial}{\partial \theta_0} (\theta_0 + \theta_1 x^{(i)} - y^{(i)}) = 1$$

GD Algorithm (Reviewed)

Then, the partial derivatives work like this:

$$\frac{\partial}{\partial\theta_0} g(f(\theta_0,\theta_1)^{(i)}) = \frac{\partial}{\theta_0} g(\theta_0,\theta_1) \cdot \frac{\partial}{\partial\theta_0} f(\theta_0,\theta_1)^{(i)} \qquad \text{(by the chain rule)}$$

$$= \frac{1}{m}\Sigma_{i=1}^{m} f(\theta_0,\theta_1)^{(i)} \times 1$$

$$= \frac{1}{m}\Sigma_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)}) \times 1 \quad = \frac{1}{m}\Sigma_{i=1}^{m} (\theta_0 + \theta_1 x^{(i)} - y^{(i)})$$

Solving for $\frac{\partial}{\partial\theta_1} J(\theta_0,\theta_1)$ is remained for a homework !

GD Algorithm (Reviewed)

repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \Sigma_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \Sigma_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

}

Update θ_0 and θ_1 simultaneously

Alternatives of GD

Two widely-used alternatives of batch gradient descent are:

- Stochastic gradient descent (SGD)
- Mini-batch gradient descent (Mini-batch)

Intuitively, ‘stochastic’ means having a random probability distribution.

The term ‘stochastic’ comes from the fact that $\nabla_J(\boldsymbol{\theta})$ is based on a single training sample that is a ‘stochastic approximation’ of the ‘true’ gradient.

Stochastic gradient descent tends to get close to the minimum faster than ‘batch’ gradient descent; but may oscillate around the minimum !

SGD Algorithm (Reviewed)

$\theta_0, \theta_1 :=$ some initial guess

repeat until convergence {

For $i \in \{1, \ldots, m\}$

$$\theta_0 := \theta_0 - \alpha(h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha(h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

}

In fact, there are several versions of SGD (*cf.* [1, 2] for references)

---

[1] Bottou, Léon (1998). "Online Algorithms and Stochastic Approximations". Online Learning and Neural Networks. Cambridge University Press. ISBN 978-0-521-65263-6
[2] Bottou, Léon. "Large-scale machine learning with SGD." Proceedings of COMPSTAT'2010. Physica-Verlag HD, 2010. 177-186.

Mini-batch GD Algorithm

$\theta_0, \theta_1 :=$ some initial guess

repeat until convergence {

let $k \in \{1, \ldots, m\}$

$$\theta_0 := \theta_0 - \alpha \frac{1}{k} \Sigma_{i=1}^{k} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{k} \Sigma_{i=1}^{k} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

}

SGD *vs.* Mini-batch

- In general, SGD can avoid local minimums because of the randomness of an input.
- SGD may take longer time to converge

- Mini-batch is a compromise version of GD and SGD.
- Mini-batch can take more computational power if the batch size is large (a common batch size is 50).

Ordinary GD is often recommended when $m \leq 1{,}000$

[1] https://medium.com/coinmonks/stochastic-vs-mini-batch-training-in-machine-learning-using-tensorflow-and-python-7f9709143ee2
[2] https://sebastianraschka.com/faq/docs/closed-form-vs-gd.html

Now, you are ready for the lab !