# 9. Gradient Descent and Variants

# Gradient Descent: Context

## Machine Learning Problems

Many machine learning problems involve the minimization of some cost function $J(\boldsymbol{\theta})$ with respect to some underlying model parameters $\boldsymbol{\theta}$, leading to

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} J(\boldsymbol{\theta})$$

## Example: Linear Regression

In simple linear regression, the cost function is given by:

$$J(\boldsymbol{\theta}) = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - \boldsymbol{\theta}^t \boldsymbol{x}_i)^2$$

There is a closed-form solution for the optimal parameters

## Example: Logistic Regression

In logistic regression, the cost function is given by:

$$J(\boldsymbol{\theta}) = -\frac{1}{n} \cdot \sum_{i=1}^{n} \mathrm{y_i} \cdot \log_2\big(\Pr(\hat{y} = 1|\boldsymbol{x}_i; \boldsymbol{\theta})\big) + (1 - y_i) \cdot \log_2\big(1 - \Pr(\hat{y} = 1|\boldsymbol{x}_i; \boldsymbol{\theta})\big)$$

There is **NO** closed-form solution for the optimal parameters

# Gradient Descent

## Gradient Descent

Gradient descent solves problems of the form:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} J(\theta)$$

It is an iterative algorithm involving two main steps:

Step1: It starts with an initial value of the parameters' vector

Step2: It updates the parameters' vector iteratively using the rule

$$\theta_i^{(j)} = \theta_i^{(j-1)} - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta_i}, i = 0, \dots, m$$

where $\theta_i^{(j)}$ is the value of the *i*-th parameter at iteration *j* and $\alpha$ is the learning rate.

**Gradient Descent Algorithm (Pseudocode)**

Repeat {

$$temp0 := \theta_0 - \alpha \cdot \partial J(\boldsymbol{\theta})/\partial \theta_0$$
$$temp1 := \theta_1 - \alpha \cdot \partial J(\boldsymbol{\theta})/\partial \theta_1$$
$$temp2 := \theta_2 - \alpha \cdot \partial J(\boldsymbol{\theta})/\partial \theta_2$$
$$\theta_0 = temp0$$
$$\theta_1 = temp1$$
$$\theta_2 = temp2$$

} until convergence

Problem involving two features

Simultaneous update

# Gradient Descent

## Gradient Descent

Gradient descent solves problems of the form:

$$\theta^* = \underset{\theta}{\operatorname{argmin}} J(\theta)$$

Derivative term

Learning rate

It is an iterative algorithm involving two main steps:

Step1: It starts with an initial value of the parameters' vector

Step2: It updates the parameters' vector iteratively using the rule

$$\theta_i^{(j)} = \theta_i^{(j-1)} - \alpha \cdot \frac{\partial J(\theta)}{\partial \theta_i}, i = 0, \dots, m$$

where $\theta_i^{(j)}$ is the value of the *i*-th parameter at iteration *j* and $\alpha$ is the learning rate.

**Gradient Descent Algorithm (Pseudocode)**

Repeat {

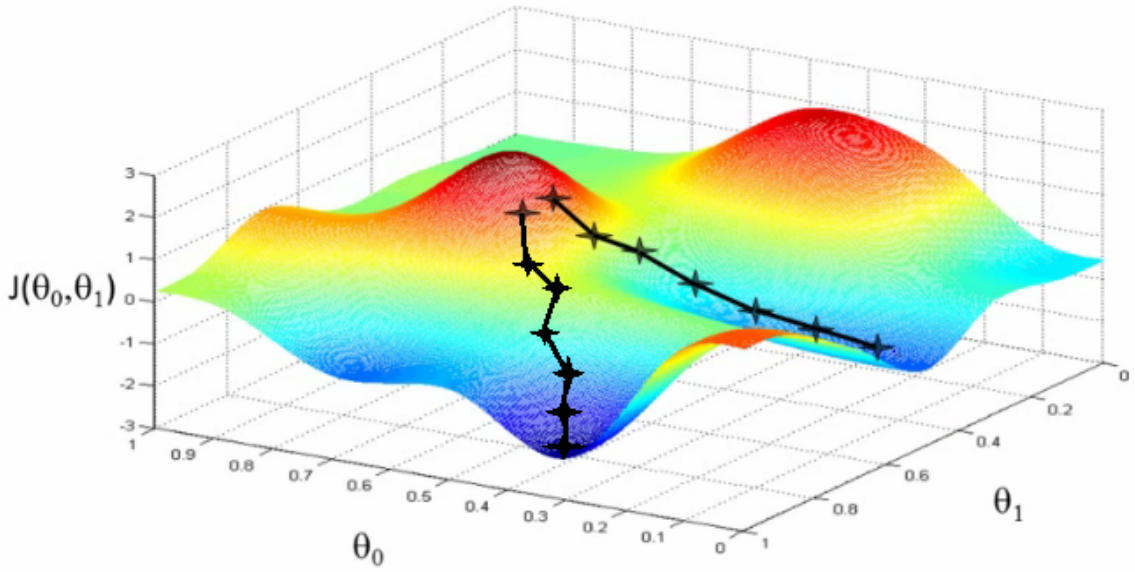
$$temp0 := \theta_0 - \alpha \cdot \partial J(\boldsymbol{\theta})/\partial \theta_0$$
$$temp1 := \theta_1 - \alpha \cdot \partial J(\boldsymbol{\theta})/\partial \theta_1$$
$$temp2 := \theta_2 - \alpha \cdot \partial J(\boldsymbol{\theta})/\partial \theta_2$$
$$\theta_0 = temp0$$
$$\theta_1 = temp1$$
$$\theta_2 = temp2$$

} until convergence

Problem involving two features

# Gradient Descent: Interpretation

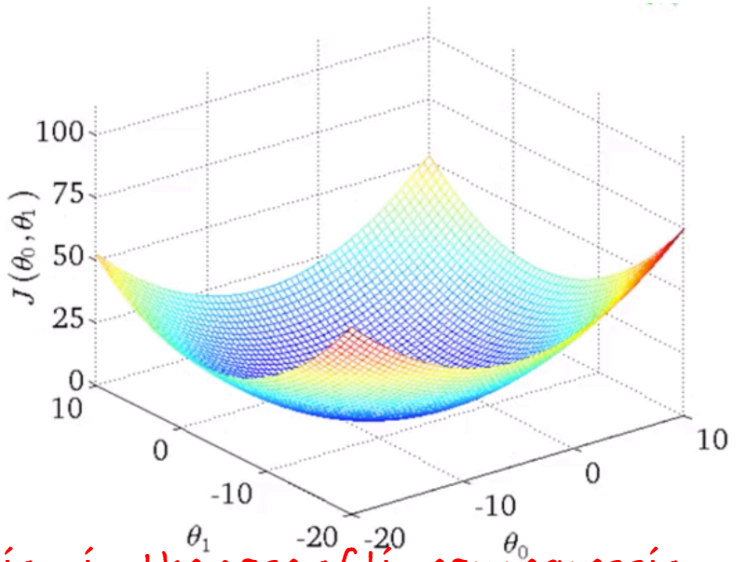
## Gradient Descent Operation




Andrew Ng, Machine Learning

## Gradient Descent Issues

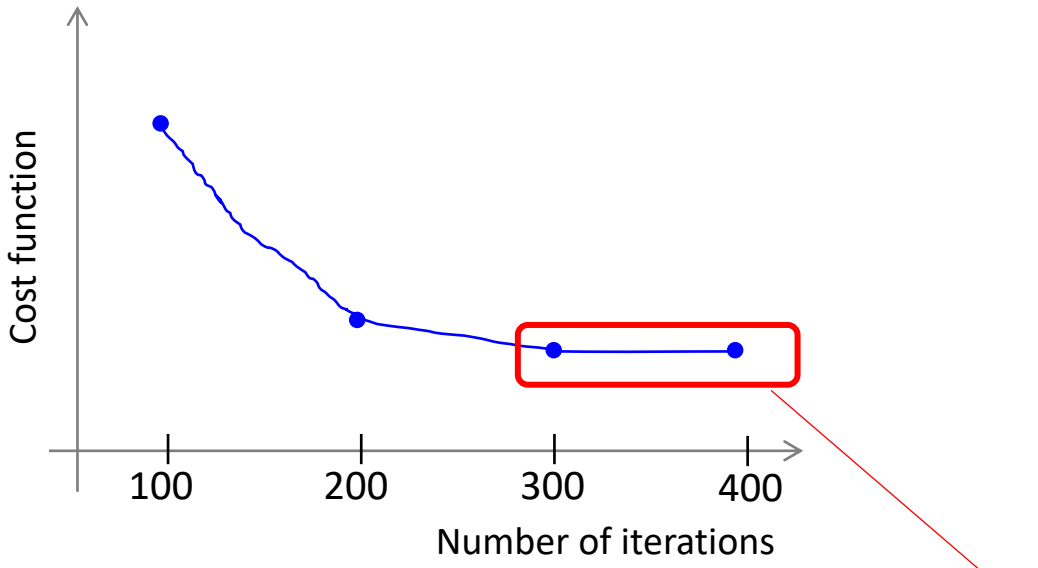
- Gradient descent can be trapped to local minima associated with the cost function
- Gradient descent is also susceptible to different initializations




Convex cost function in the case of linear regression

# Gradient Descent: Considerations (1)

## Convergence Tests



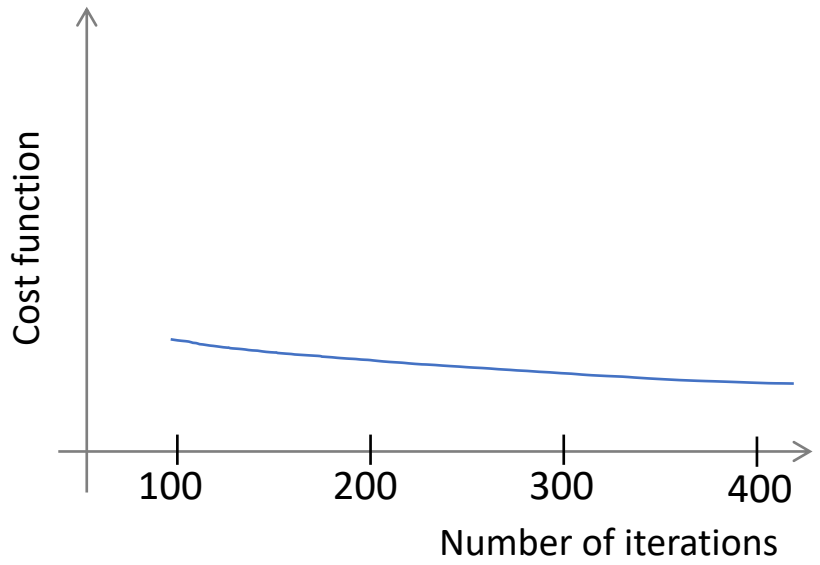

- The cost function decreases by less than a threshold
- The cost function does not change by more than a threshold from one iteration to the next
- The parameter values do not change by more than a threshold from one iteration to the next

GD has converged

Decreasing behavior → good GD

# Gradient Descent: Considerations (2)

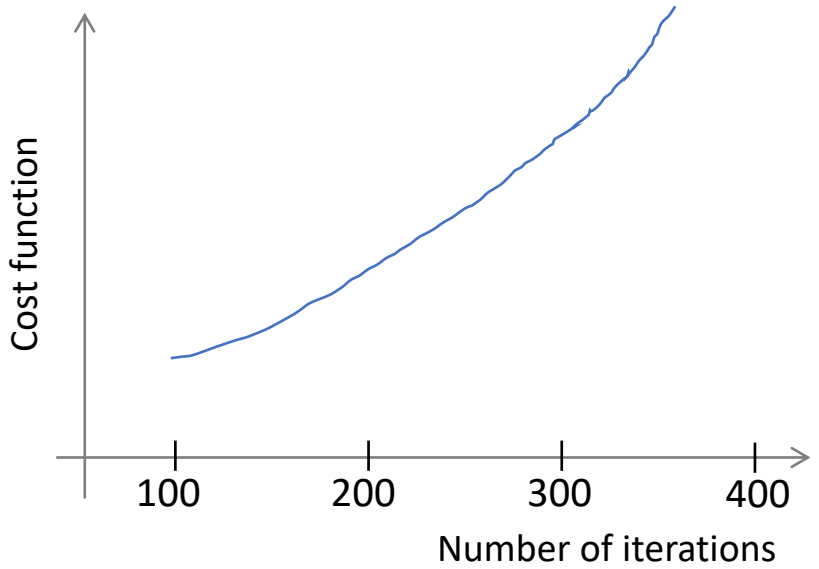
## Small Learning Rate $\alpha$




### Implications

The algorithm converges but very slowly

## Large Learning Rate $\alpha$




### Implications

The algorithm may not converge

Rule of Thumb:
Try 0.001, 0.003, 0.01, 0.1, 1

# Gradient Descent: Feature Normalization

(特征缩放)

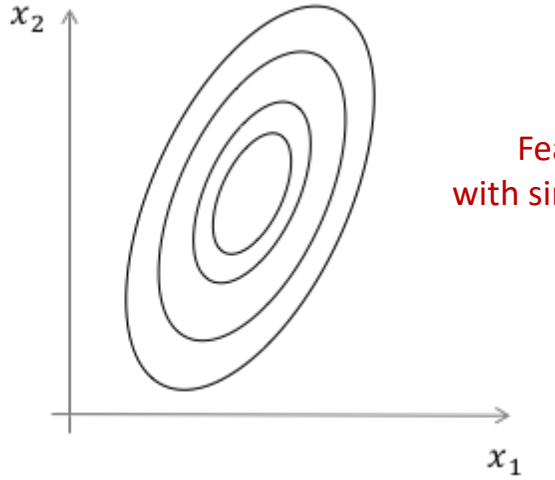
## Feature Normalization

Some machine learning problems involve features with completely different scales.

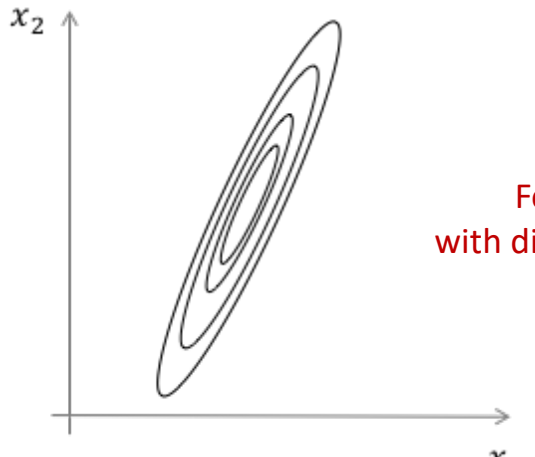
Example the prediction of second-hand car price problem involves features such as $x_1$ = mileage (10.000 – 100.000 km) and $x_2$ = number of doors (2-5)

It is important features to have similar scales otherwise gradient descent can take long to converge

## Contour Plots




Features with similar scale




Features with different scale

# Gradient Descent: Feature Normalization

## Feature Normalization

Some machine learning problems involve features with completely different scales.

Example the prediction of second-hand car price problem involves features such as $x_1$ = mileage (10.000 – 100.000 km) and $x_2$ = number of doors (2-5)

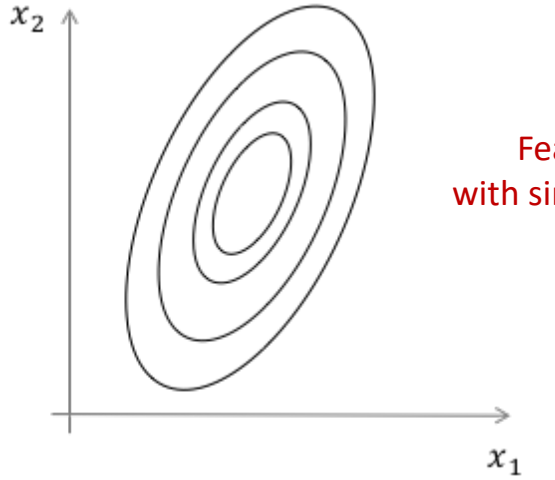
It is important features to have similar scales otherwise gradient descent can take long to converge

This can be done by applying the transformation

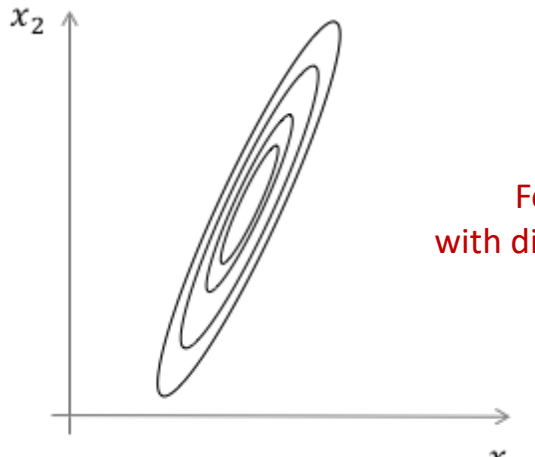
$$x_i \longrightarrow \frac{(x_i - \mu_i)}{\sigma_i}$$

where $\mu_i$ is the mean of the feature and $\sigma_i$ is its standard deviation

## Contour Plots




Features with similar scale




Features with different scale

# Gradient Descent: Feature Normalization

## Feature Normalization

Some machine learning problems involve features with completely different scales.

Example the prediction of second-hand car price problem involves features such as $x_1$ = mileage (10.000 – 100.000 km) and $x_2$ = number of doors (2-5)

It is important features to have similar scales otherwise gradient descent can take long to converge

This can be done by applying the transformation

$$x_i \longrightarrow \frac{(x_i - \mu_i)}{\sigma_i}$$

where $\mu_i$ is the mean of the feature and $\sigma_i$ is its standard deviation

Rule of Thumb:

Features within the ranges

[-3,3]
[-1/3,1/3]

Are overall fine.

# Gradient Descent in Linear Regression

## Cost Function

The cost function associated with simple linear regression is given by:

$$J(\boldsymbol{\theta}) = \frac{1}{2 \cdot n} \sum_{i=1}^{n} (y_i - \boldsymbol{\theta}^t \boldsymbol{x}_i)^2$$

## Gradient Descent Update Rules

The gradient descent update rules are given by:

$$\theta_k^{(j)} = \theta_k^{(j-1)} - \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{\theta}^t \boldsymbol{x}_i - y_i) \cdot x_{i,k}$$

### Gradient Descent for Linear Regression

Repeat {

$$temp0 := \theta_0 - \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{\theta}^t \boldsymbol{x}_i - y_i) \cdot x_{i,0}$$

$$temp1 := \theta_1 - \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{\theta}^t \boldsymbol{x}_i - y_i) \cdot x_{i,1}$$

$$temp2 := \theta_2 - \alpha \cdot \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{\theta}^t \boldsymbol{x}_i - y_i) \cdot x_{i,2}$$

$$\theta_0 = temp0$$

$$\theta_1 = temp1$$

$$\theta_2 = temp2$$

} until convergence

Problem involving two features

# Gradient Descent in Logistic Regression

## Cost Function

The cost function associated with simple linear regression is given by:

$$J(\boldsymbol{\theta}) = -\frac{1}{n}\sum_{i=1}^{n} \mathrm{y_i} \cdot \log_2\left(\Pr(\hat{y} = 1|\boldsymbol{x}_i; \boldsymbol{\theta})\right) + (1 - y_i) \cdot \log_2\left(1 - \Pr(\hat{y} = 1|\boldsymbol{x}_i; \boldsymbol{\theta})\right)$$

## Gradient Descent Update Rules

The gradient descent update rules are given by:

$$\theta_k^{(j)} = \theta_k^{(j-1)} - \alpha \cdot \frac{1}{\ln 2} \cdot \frac{1}{n}\sum_{i=1}^{n} (g(\boldsymbol{\theta}^t \boldsymbol{x_i}) - y_i) \cdot x_{i,k}$$

### Gradient Descent for Linear Regression

Repeat {

$$temp0 \coloneqq \theta_0 - \alpha \cdot \frac{1}{\ln 2} \cdot \frac{1}{n}\sum_{i=1}^{n} (g(\boldsymbol{\theta}^t \boldsymbol{x}_i) - y_i) \cdot x_{i,0}$$

$$temp1 \coloneqq \theta_1 - \alpha \cdot \frac{1}{\ln 2} \cdot \frac{1}{n}\sum_{i=1}^{n} (g(\boldsymbol{\theta}^t \boldsymbol{x}_i) - y_i) \cdot x_{i,1}$$

$$temp2 \coloneqq \theta_2 - \alpha \cdot \frac{1}{\ln 2} \cdot \frac{1}{n}\sum_{i=1}^{n} (g(\boldsymbol{\theta}^t \boldsymbol{x}_i) - y_i) \cdot x_{i,2}$$

$$\theta_0 = temp0$$

$$\theta_1 = temp1$$

$$\theta_2 = temp2$$

} until convergence

Problem involving two features

# Gradient Descent for Softmax Regression

## Cost Function

The cost function associated with simple linear regression is given by:

$$J(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K} 1\{y^{(i)} = k\} \cdot \log_2 \frac{e^{\boldsymbol{\theta}_k^t \cdot x^{(i)}}}{\sum_{j=1}^{K} e^{\theta_j^t \cdot x^{(i)}}}$$

## Gradient Descent Update Rules

The gradient descent update rules are given by:

$$\boldsymbol{\theta}_k^{(j)} = \boldsymbol{\theta}_k^{(j-1)} - \alpha \cdot \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}^{(i)} \cdot \left(1\{y^{(i)} = k\} - Pr\left(y^{(i)} = k | \boldsymbol{x}^{(i)}; \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K\right)\right)$$

# Does Gradient Descent Scale with the Size of the Training Set?

# Stochastic Gradient Descent

## Cost Function

Individual cost per training sample

$$\underbrace{J(\boldsymbol{\theta}) = \frac{1}{n} \cdot \sum_{i=1}^{n} \overbrace{\text{cost}\big(\boldsymbol{\theta}; (\boldsymbol{x}_i, y_i)\big)}}$$

Average cost over all training samples

In the case of linear regression

$$\text{cost}\big(\boldsymbol{\theta}; (\boldsymbol{x}_i, y_i)\big) = \frac{1}{2}(y_i - \boldsymbol{\theta}^t \boldsymbol{x}_i)^2$$

# Stochastic Gradient Descent

## Cost Function

Individual cost per training sample

$$J(\boldsymbol{\theta}) = \frac{1}{n} \cdot \sum_{i=1}^{n} \overbrace{\text{cost}\big(\boldsymbol{\theta}; (\boldsymbol{x}_i, y_i)\big)}$$

Average cost over all training samples

## Gradient Descent

Repeat {

$$\boldsymbol{\theta}_i^{(j)} = \boldsymbol{\theta}_i^{(j-1)} - \alpha \cdot \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_i}, i = 0, \dots, m$$

} until convergence

# Stochastic Gradient Descent

## Cost Function

Individual cost per training sample

$$J(\boldsymbol{\theta}) = \frac{1}{n} \cdot \sum_{i=1}^{n} \text{cost}\big(\boldsymbol{\theta}; (\boldsymbol{x}_i, y_i)\big)$$

Average cost over all training samples

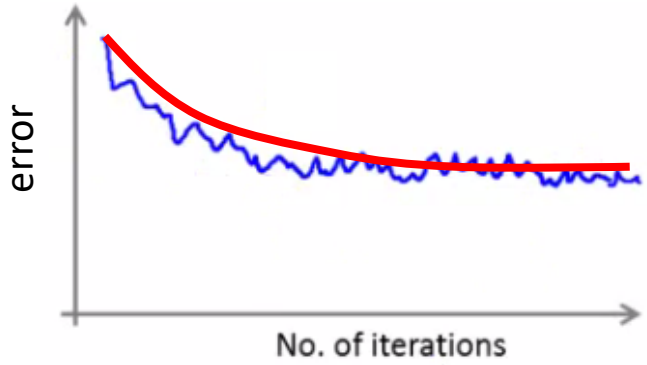
## Gradient Descent

Repeat {

$$\boldsymbol{\theta}_i^{(j)} = \boldsymbol{\theta}_i^{(j-1)} - \alpha \cdot \frac{\partial J(\boldsymbol{\theta})}{\partial \theta_i}, i = 0, \dots, m$$

} until convergence

## Convergence




Gradient descent

Stochastic gradient descent

## Stochastic Gradient Descent

Repeat { ~1x-10x

for i = 1,…,n {

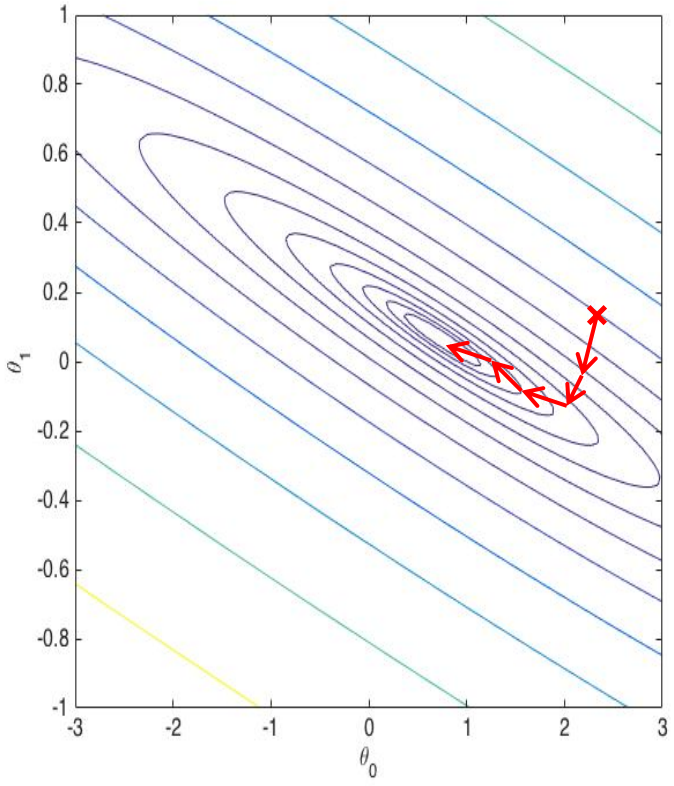
$$\boldsymbol{\theta}_i^{(j)} = \boldsymbol{\theta}_i^{(j-1)} - \alpha \cdot \frac{\partial \text{cost}\big(\boldsymbol{\theta}; (\boldsymbol{x}_i, y_i)\big)}{\partial \theta_i}, i = 0, \dots, m$$

}

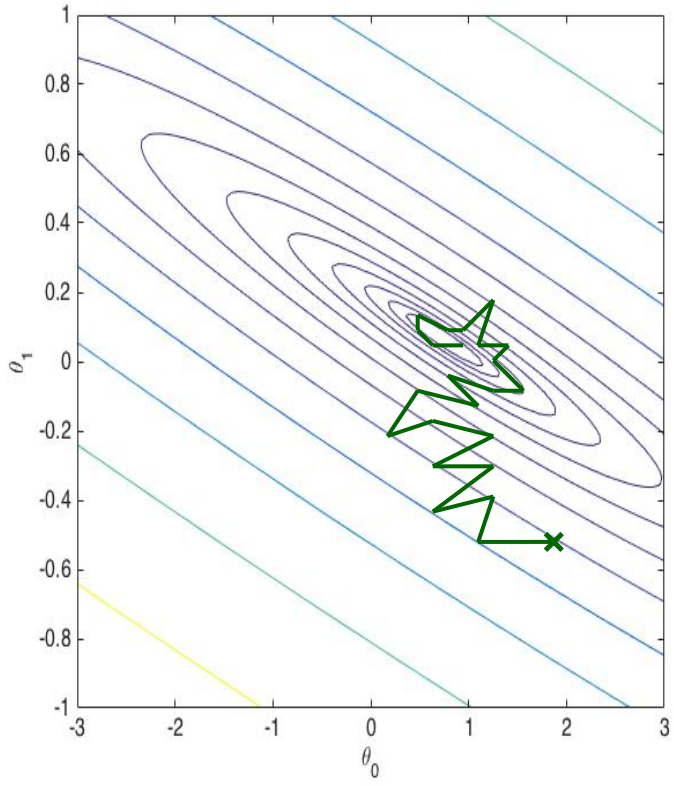
} until convergence

# Stochastic Gradient Descent

## Gradient Descent




## Stochastic Gradient Descent




It is typical to slowly decrease the learning rate over the iterations: $\alpha = C_1/(\#\text{iterations} + C_2)$

# Stochastic Gradient Descent Traits

- Stochastic gradient descent typically moves in the direction of a minimum but not always
- Stochastic gradient descent never actually converges: it wanders close to the minimum
- Stochastic gradient descent typically loops one two ten times over the entire dataset but a single pass can be entirely enough over a massive dataset
- Stochastic gradient descent and gradient descent are specific forms of **mini-batch gradient descent**.

  - In gradient descent one uses all the training data examples per iteration
  - In stochastic gradient descent one uses a single example per iteration
  - In mini-batch gradient descent one uses a batch of b examples per iteration.