# Linear Regression

Data Science Dojo

datasciencedojo
unleash the data scientist in you

# Overview

- Difference between Regression and Classification
- Linear Regression
  - Motivating Example: Predicting Housing Prices
  - Hypothesis function
  - Cost Function
  - Gradient Descent
  - Probabilistic Interpretation
- Applying linear regression for hand digit recognition

# Supervised Learning

- Data: $D = \{ d_1, d_2, d_3, \dots\dots\dots, d_n \}$ a set of n examples

  $d_i = < x_i, y_i >$

$x_i$

- Input vector
- Independent variables
- Explanatory variables
- Features
- Predictors

$y_i$

- Output scalar
- Dependent variables
- Response
- Outcome

datasciencedojo

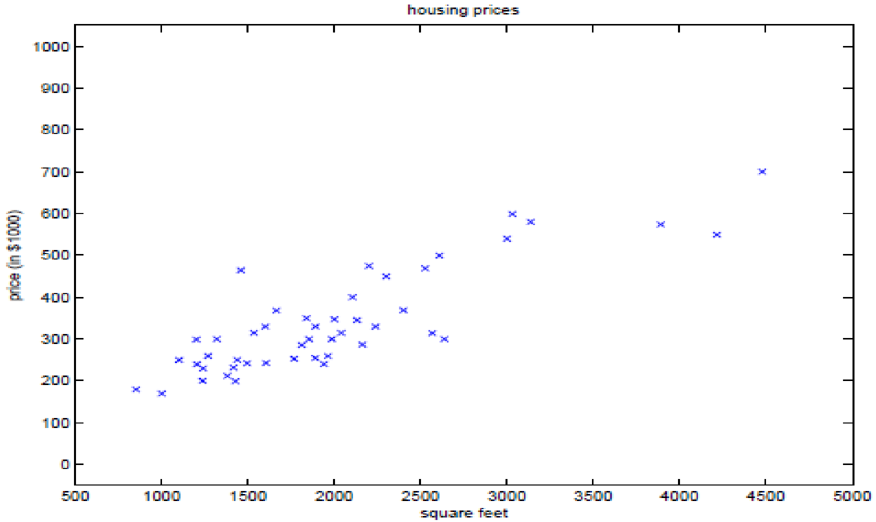
unleash the data scientist in you

# Supervised Learning

- Regression: X discrete or continuous $\longrightarrow$ Y is continuous
  - Example: Prices, Weight, Height, signal measurement, temperature etc.
- Classification : X discrete or continuous $\longrightarrow$ Y is discrete
- Objective: learn the mapping of f : Xi $\longrightarrow$ Yi

# Predict Housing Prices

| Living area ($feet^2$) | Price (1000$s) |
|---|---|
| 2104 | 400 |
| 1600 | 330 |
| 2400 | 369 |
| 1416 | 232 |
| 3000 | 540 |
| ⋮ | ⋮ |



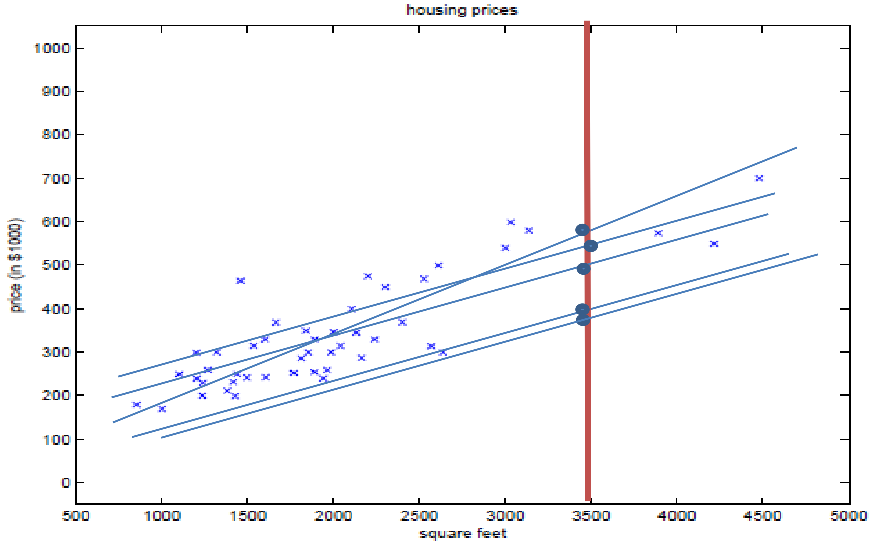

Can we learn to predict the price when the price of the house is not in the dataset?

**Living Area = 3500 sq. ft.** **Price = ?**

datasciencedojo
unleash the data scientist in you

# Predict Housing Prices

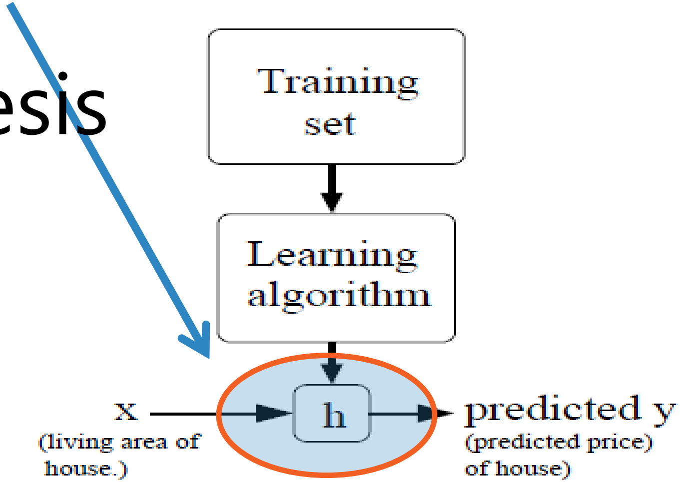
- Can we learn to predict the price when the price of the house is not in the dataset?
- Living Area = 3500 sq. ft. Price = ?
- Use the line that is somewhere in the middle
- How do we define somewhere in the middle?




datasciencedojo
unleash the data scientist in you

# Training Process In Linear Regression

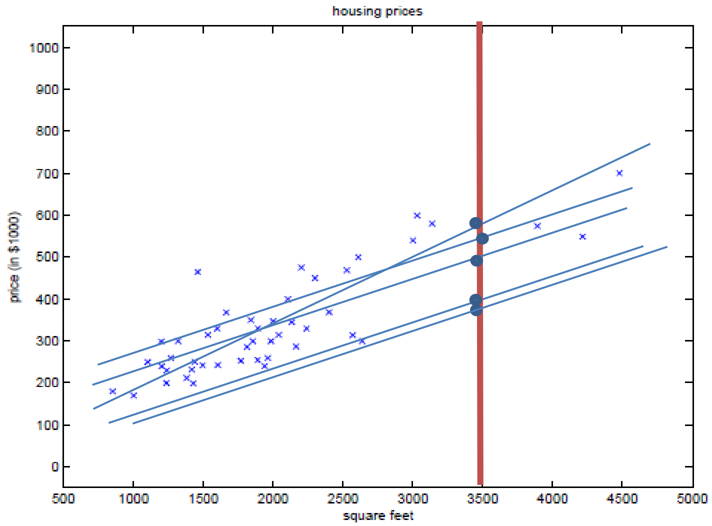
- Objective: learn the mapping of f : Xi ⟶ Yi
- Finding h is the goal here
- h is known as the hypothesis



# Predict Housing Prices

$$h_\theta(x) = \theta_0 + \theta_1 x$$

| | Living area ($\text{feet}^2$) | Price (1000$s) | |
|---|---|---|---|
| $x^{(1)}$ | 2104 | 400 | $y^{(1)}$ |
| $x^{(2)}$ | 1600 | 330 | $y^{(2)}$ |
| $x^{(3)}$ | 2400 | 369 | $y^{(3)}$ |
| $x^{(4)}$ | 1416 | 232 | $y^{(4)}$ |
| $x^{(5)}$ | 3000 | 540 | $y^{(5)}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | $\vdots$ |
| $x^{(m)}$ | | | $y^{(m)}$ |




m = Total number of training examples

$\theta_j$ = Parameters

# Hypothesis Function-Linear Regression

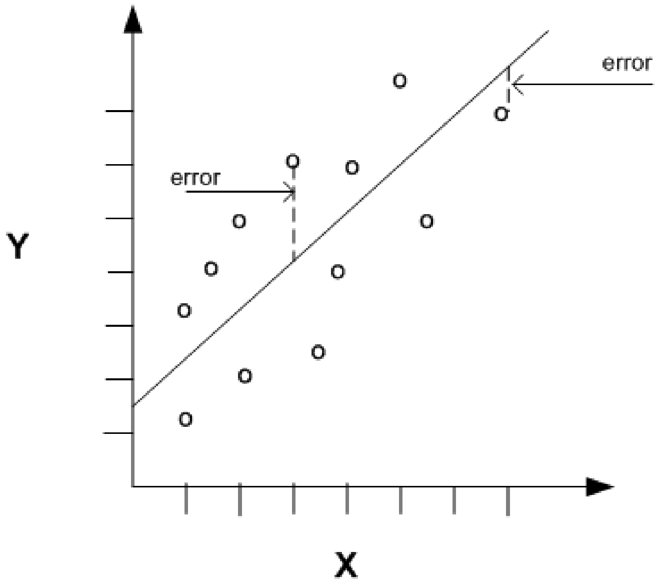
$$h_\theta(x) = \theta_0 + \theta_1 x$$

$h_\theta(x^i)$ is the prediction of the hypothesis $h_\theta$ for given input $x^i$
$y^i$ is the target values (what we would call labels in a classification problem)

We want the prediction $h_\theta(x^i)$ to be as close to the true label $y^i$ as possible.
How do we do that?

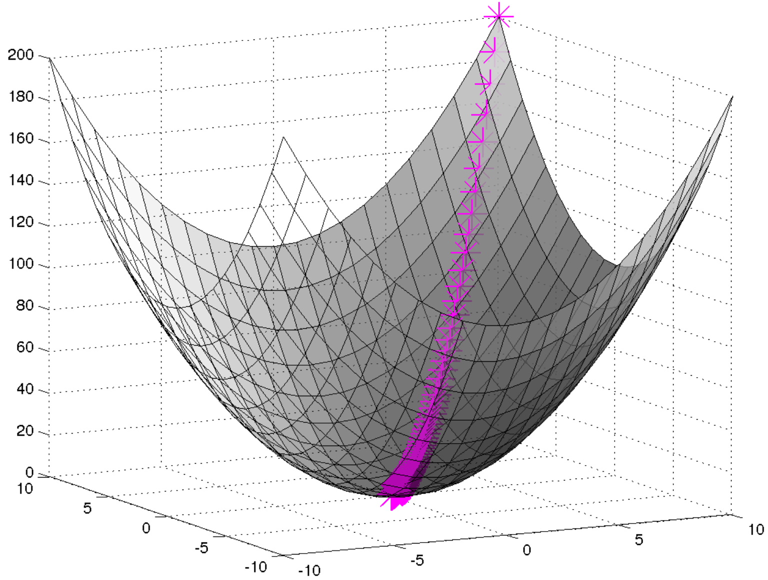
# Cost Function

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2$$




- In Plain English
  - find a linear line that is as close to the actual outputs as possible.
- Mathematically
  - Find linear function of X that minimizes the sum of squared residuals from Y.
  - a.k.a loss function

# The Cost Function




Hypothesis: $h_\theta(x) = \theta_0 + \theta_1 x$

Cost Function: $J(\theta_0, \theta_1) = \dfrac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Parameters: $\theta_0, \theta_1$

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

# Gradient Descent Algorithm

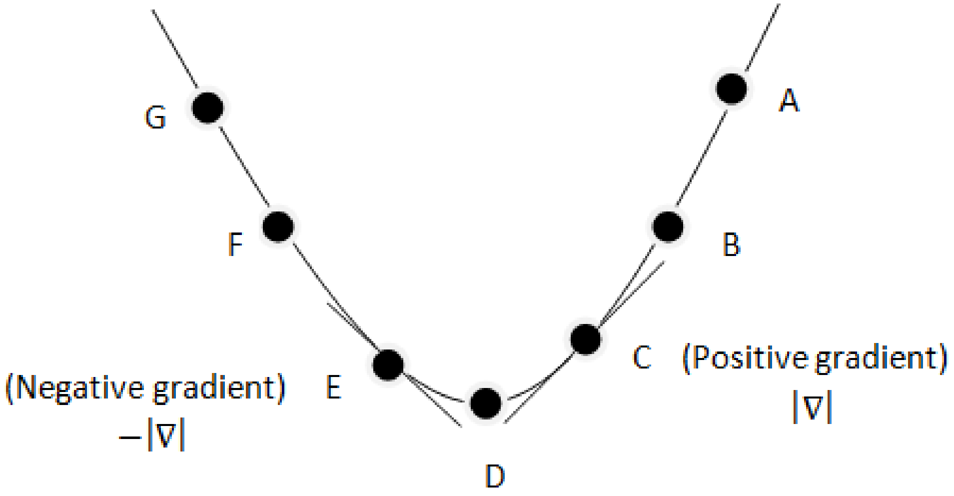
- Goal : $\underset{\theta_0,\theta_1}{\text{minimize}}\ J(\theta_0,\theta_1)$
- Gradient descent starts with some initial $\theta$ and then performs an update for each value $\theta_j$

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

- Repeat until $\theta$ converges

# Gradient Descent Algorithm

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




datasciencedojo
unleash the data scientist in you

# Gradient Descent Algorithm

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

*For j=0 and j= 1*

Repeat until converge {

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

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

}

datasciencedojo

unleash the data scientist in you

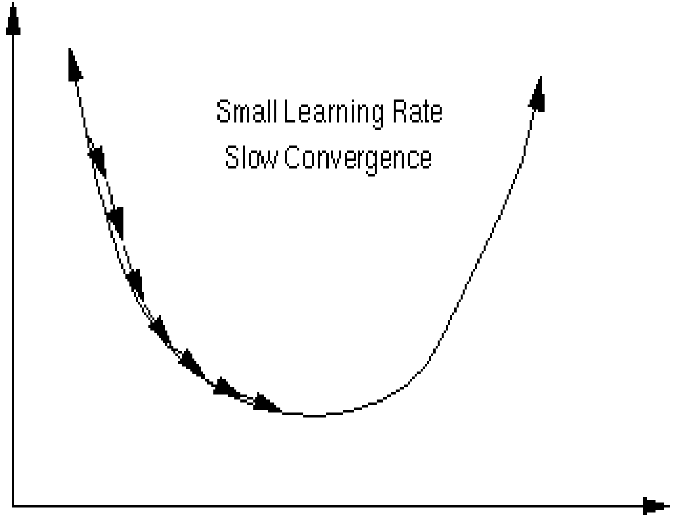
# Gradient Descent Algorithm

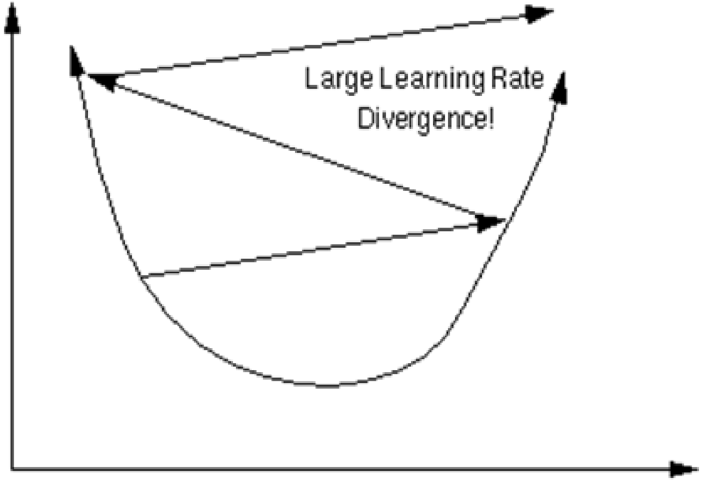
$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta).$$

- $\alpha$ is known as the learning rate
- Each time the algorithm takes a step in the direction of the steepest, $J(\theta)$ decreases.
- $\alpha$ determines how quickly or slowly the algorithm will converge to a solution

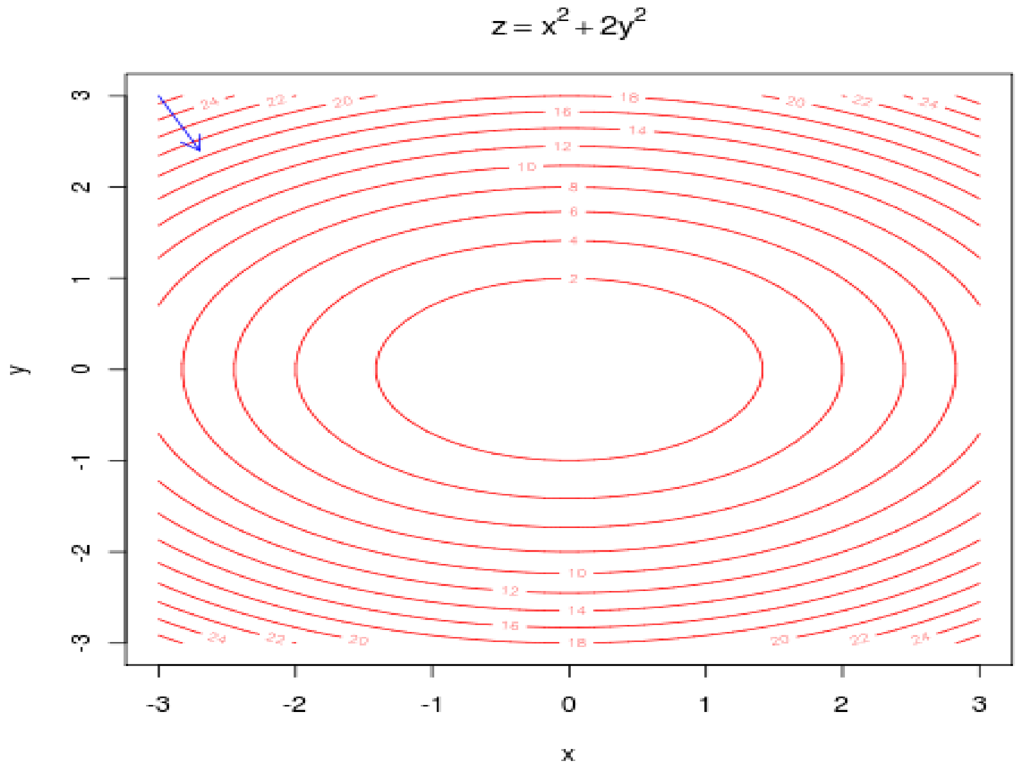
# Gradient Descent Algorithm

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

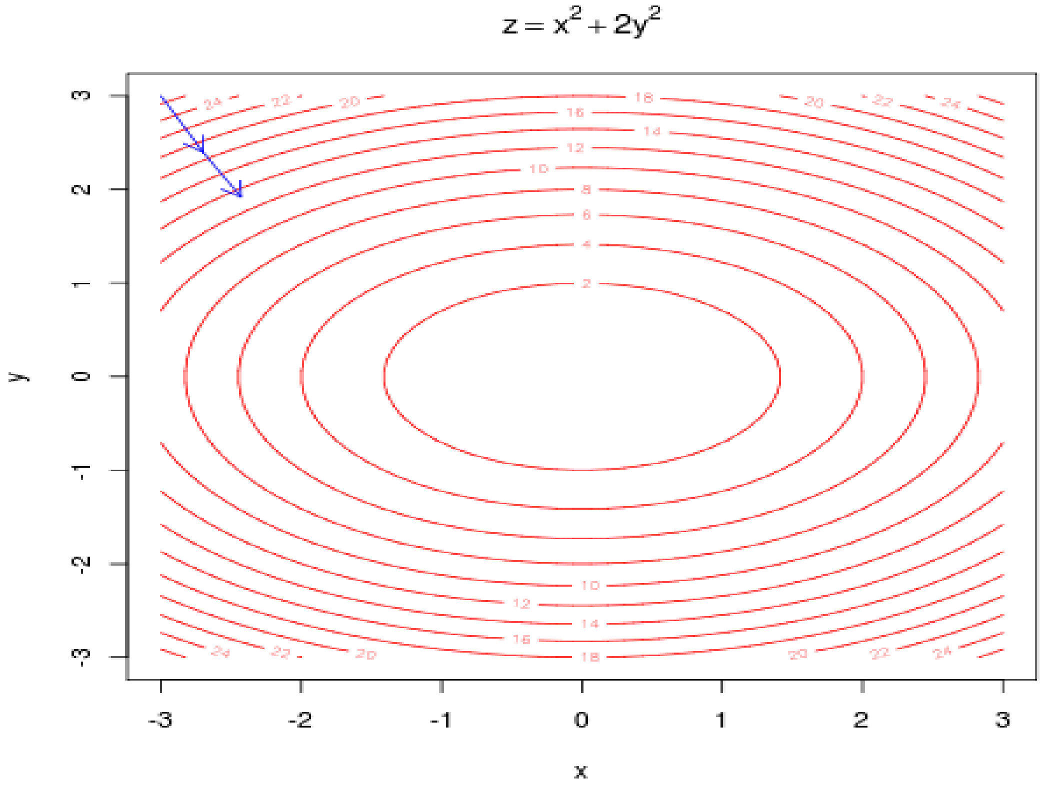


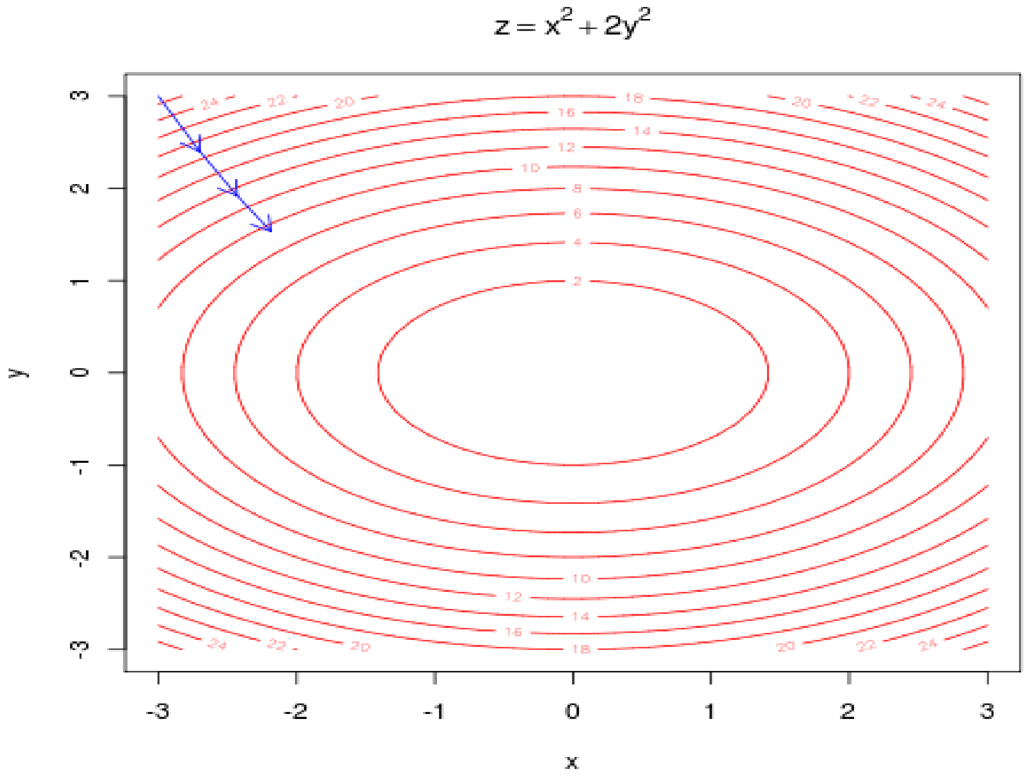
# Digression - Gradient Descent Intuition



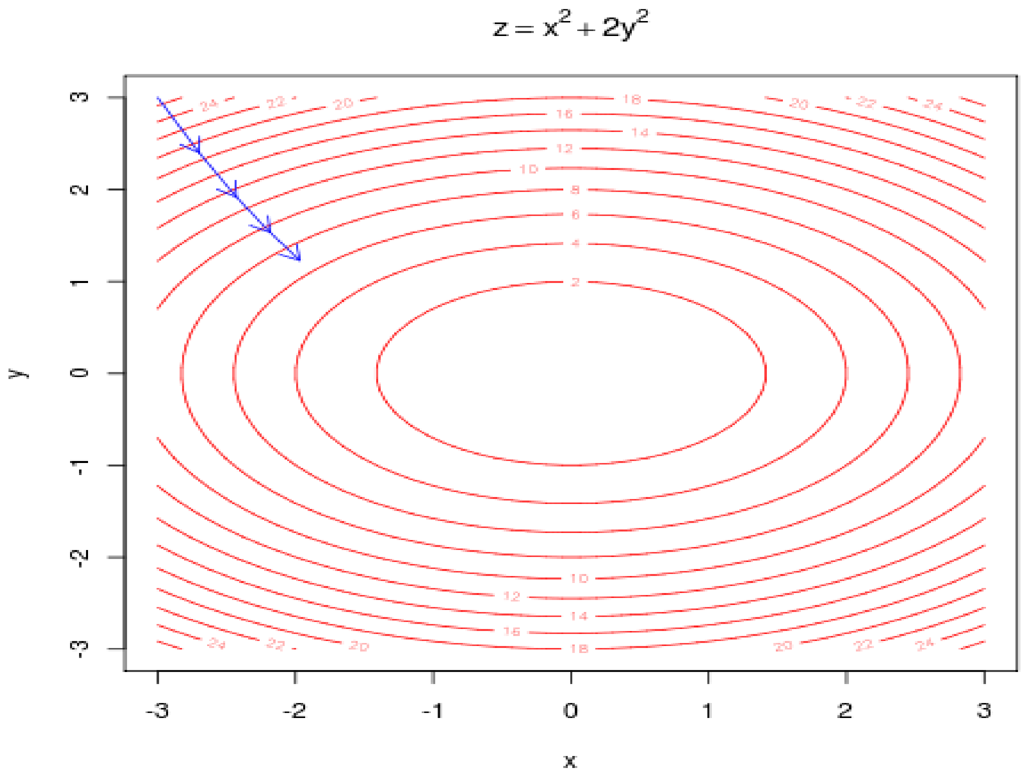
# Digression - Gradient Descent Intuition



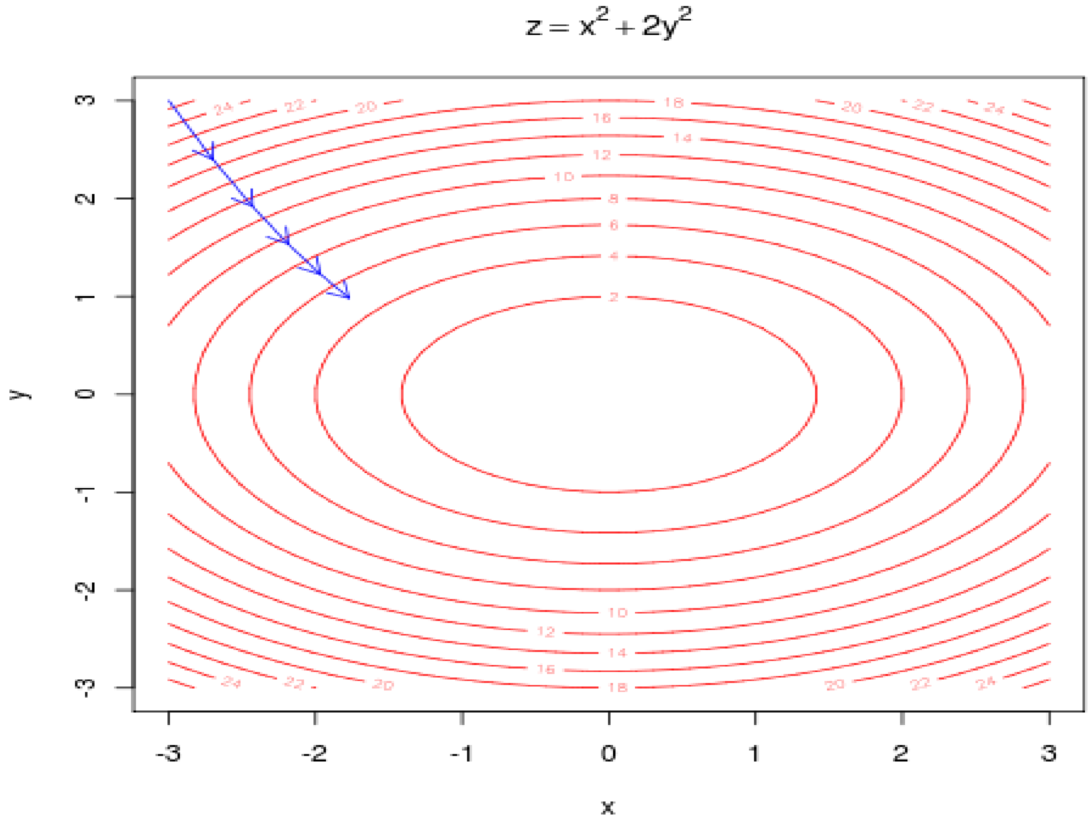
# Digression - Gradient Descent Intuition



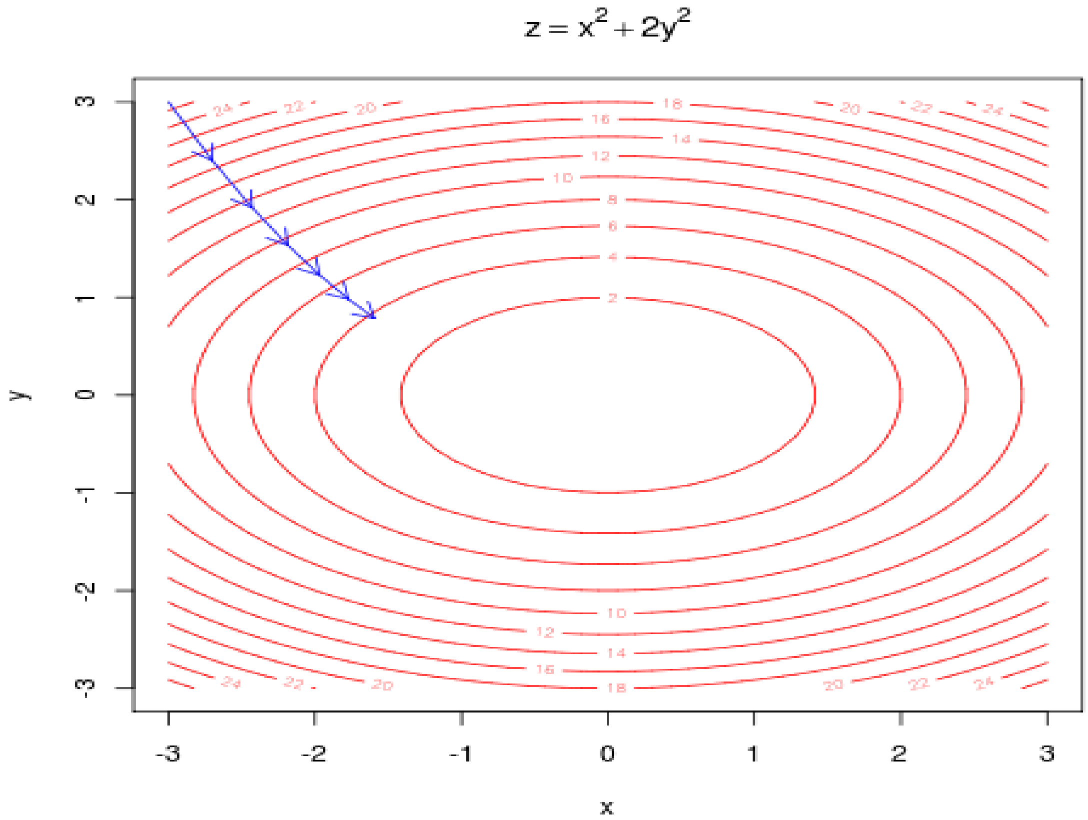
# Digression - Gradient Descent Intuition



# Digression - Gradient Descent Intuition

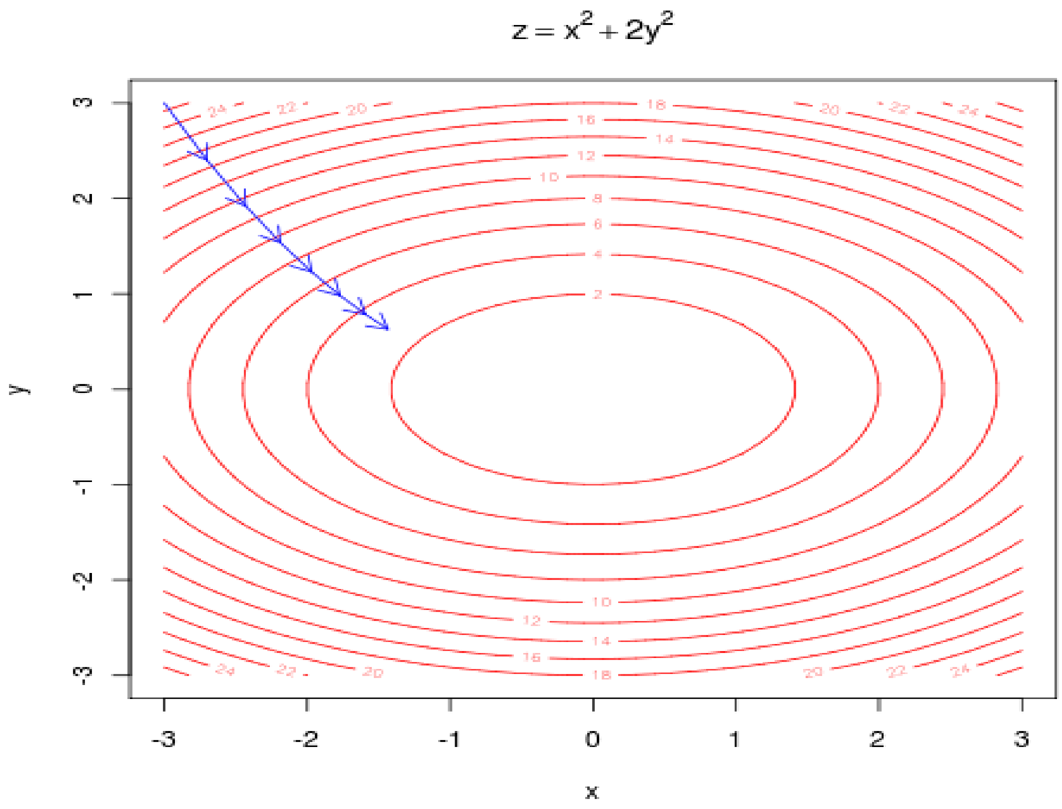


# Digression - Gradient Descent Intuition



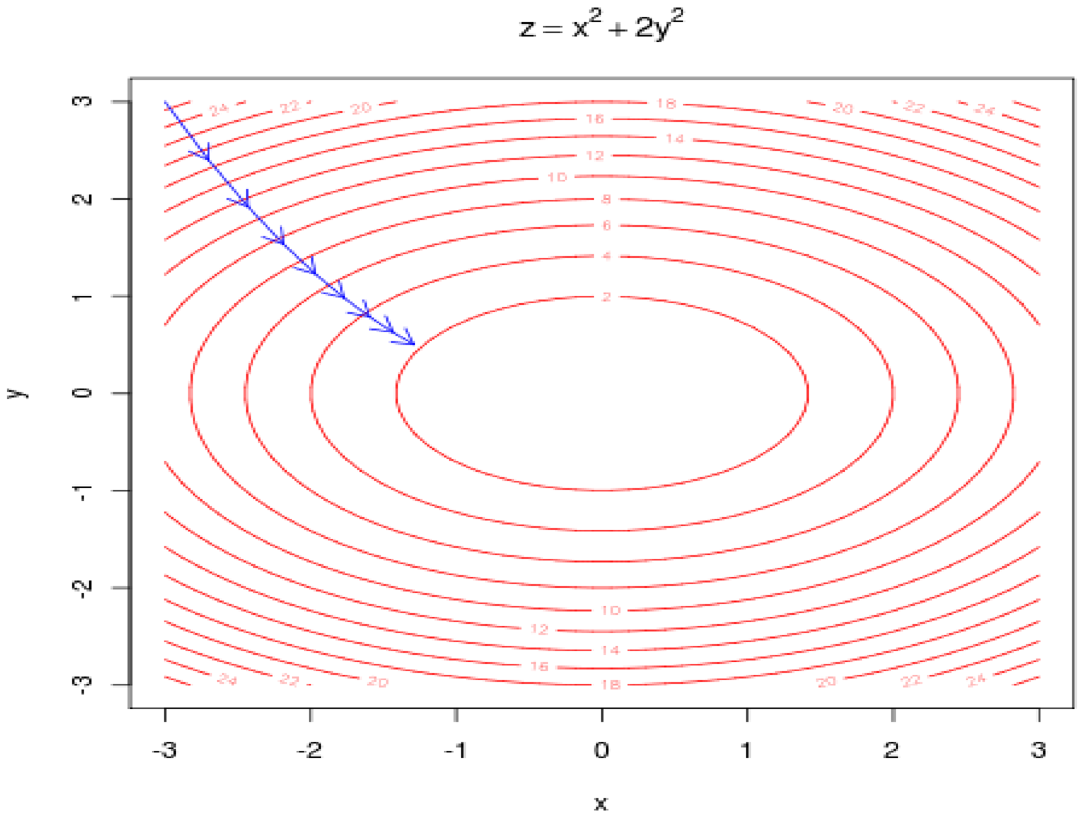

datasciencedojo
unleash the data scientist in you

# Digression - Gradient Descent Intuition



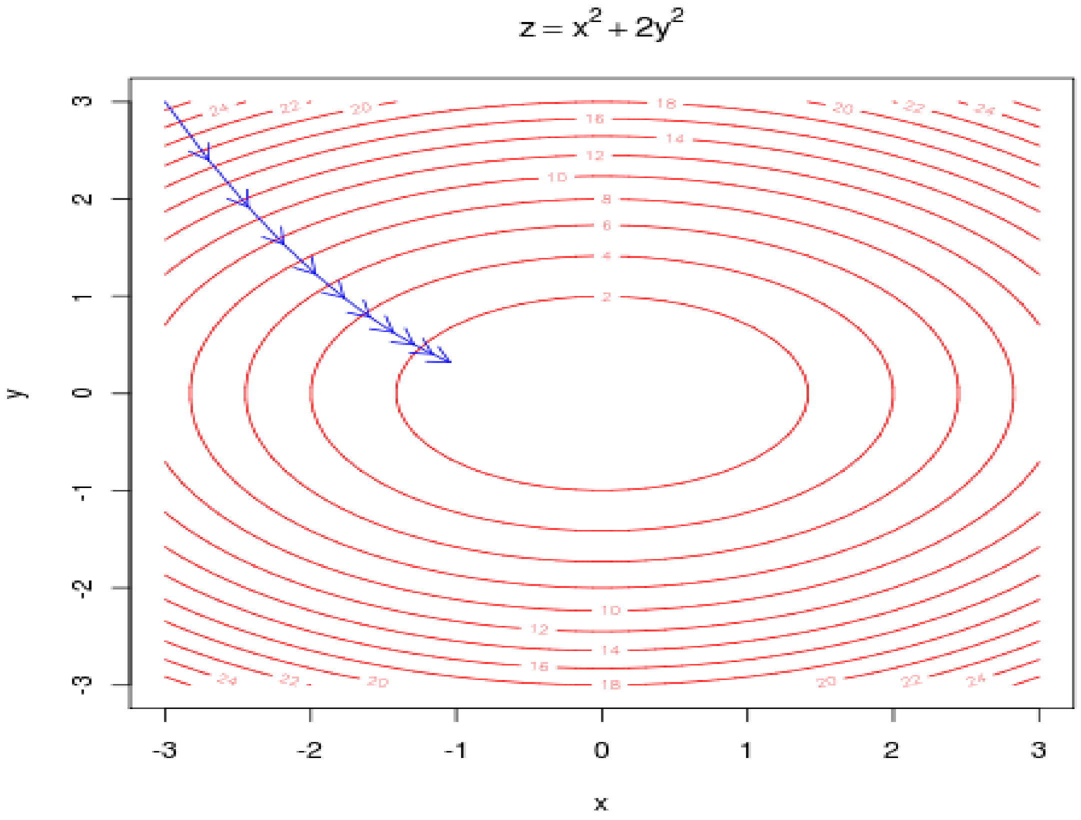

datasciencedojo
unleash the data scientist in you

# Digression - Gradient Descent Intuition



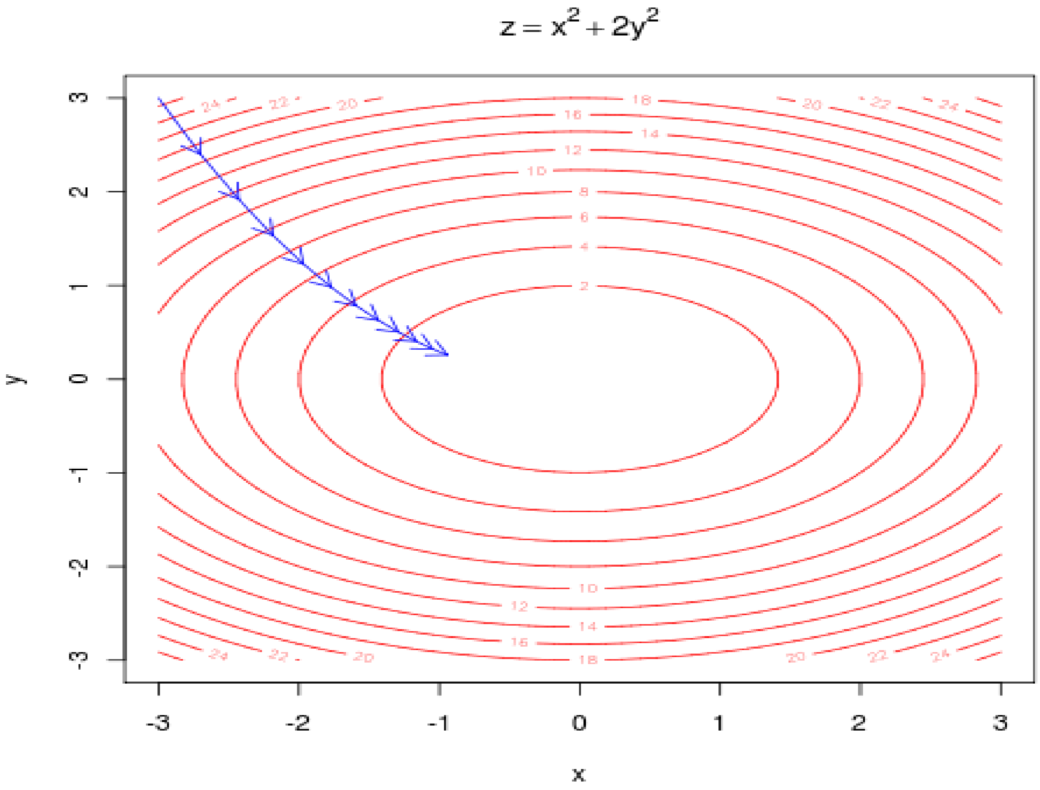

datasciencedojo
unleash the data scientist in you

# Digression - Gradient Descent Intuition



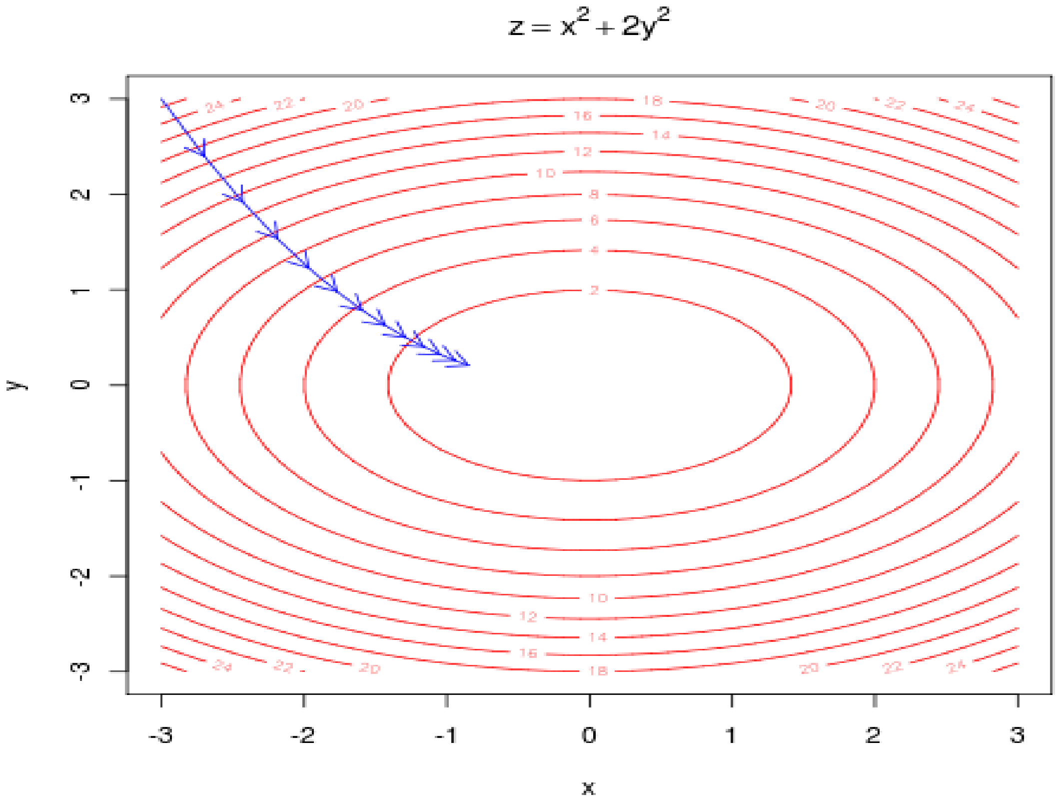

datasciencedojo
unleash the data scientist in you

# Digression - Gradient Descent Intuition



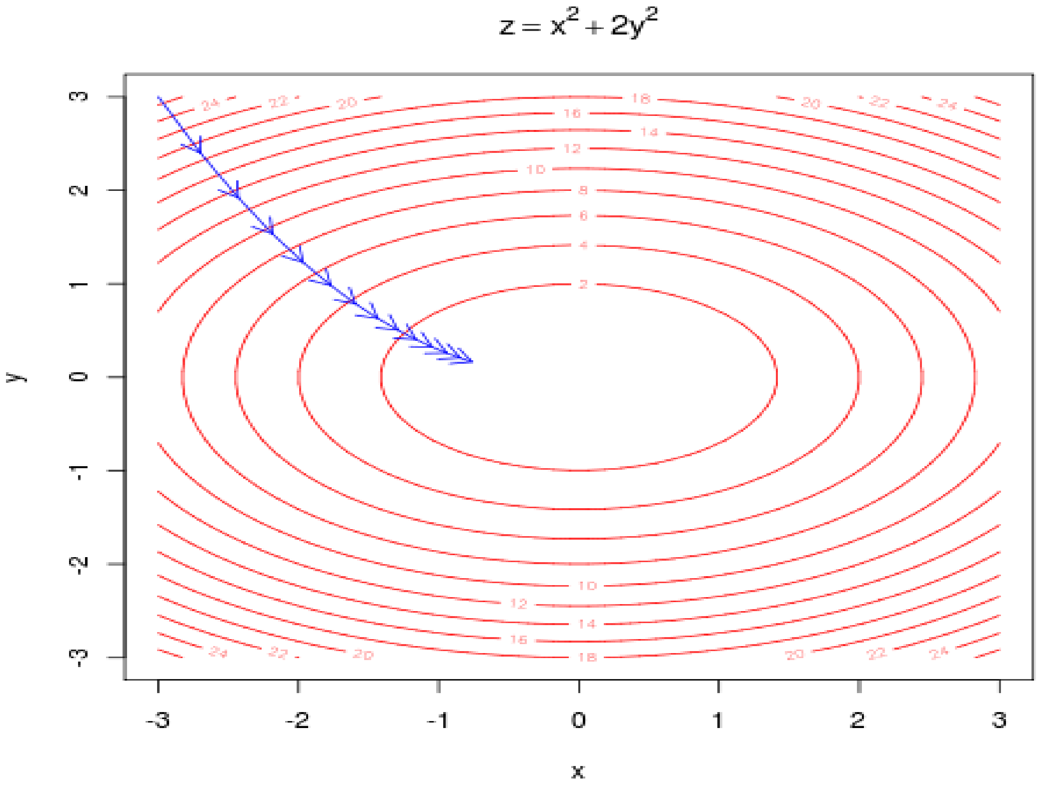

datasciencedojo
unleash the data scientist in you

# Digression - Gradient Descent Intuition



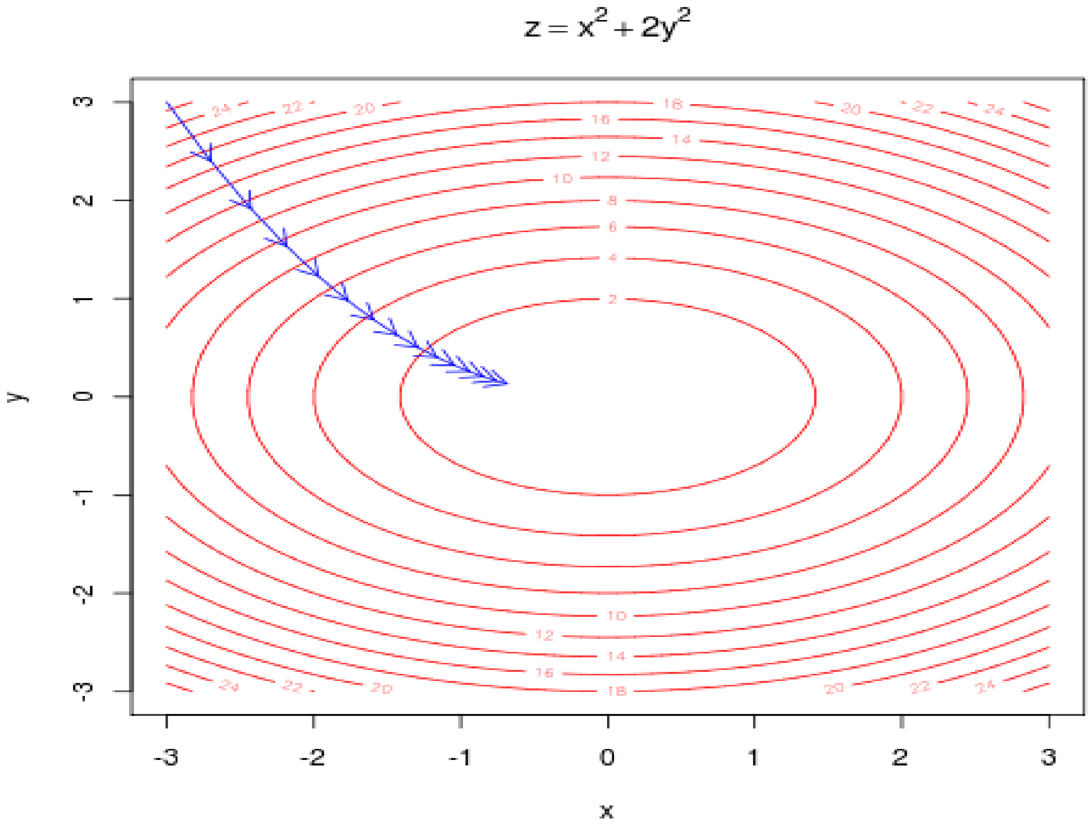

datasciencedojo
unleash the data scientist in you

# Digression - Gradient Descent Intuition

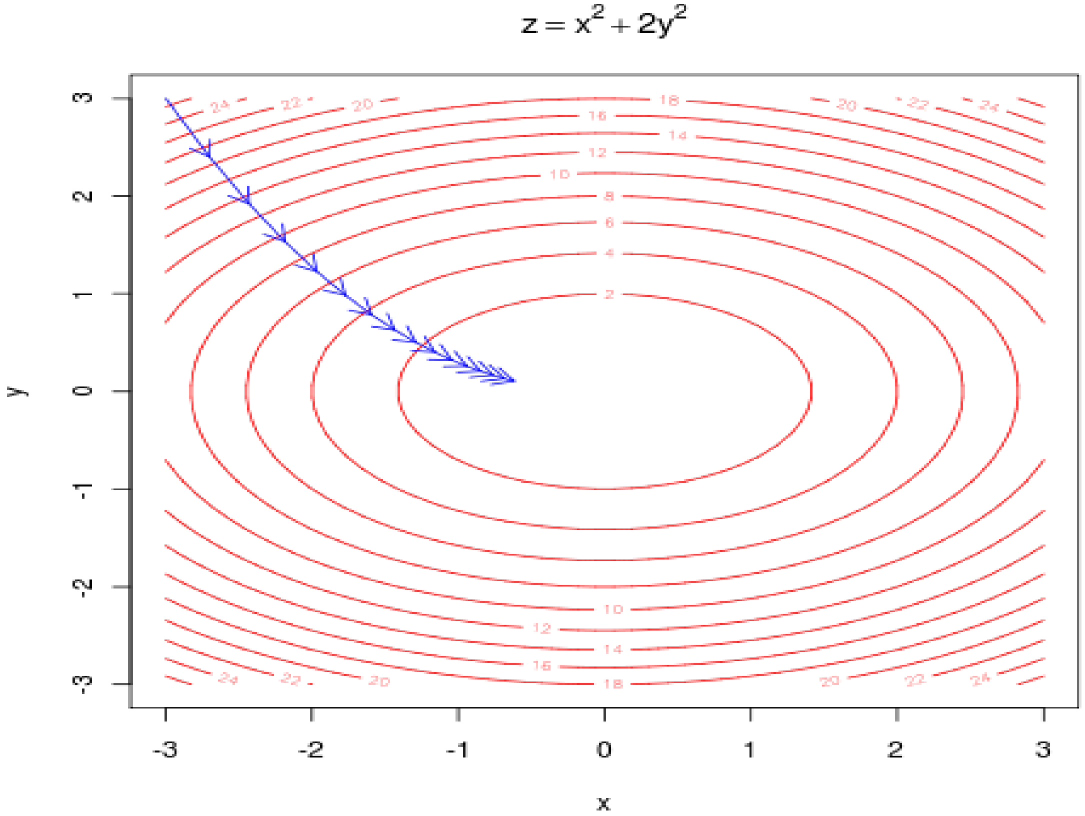



datasciencedojo
unleash the data scientist in you

# Digression - Gradient Descent Intuition

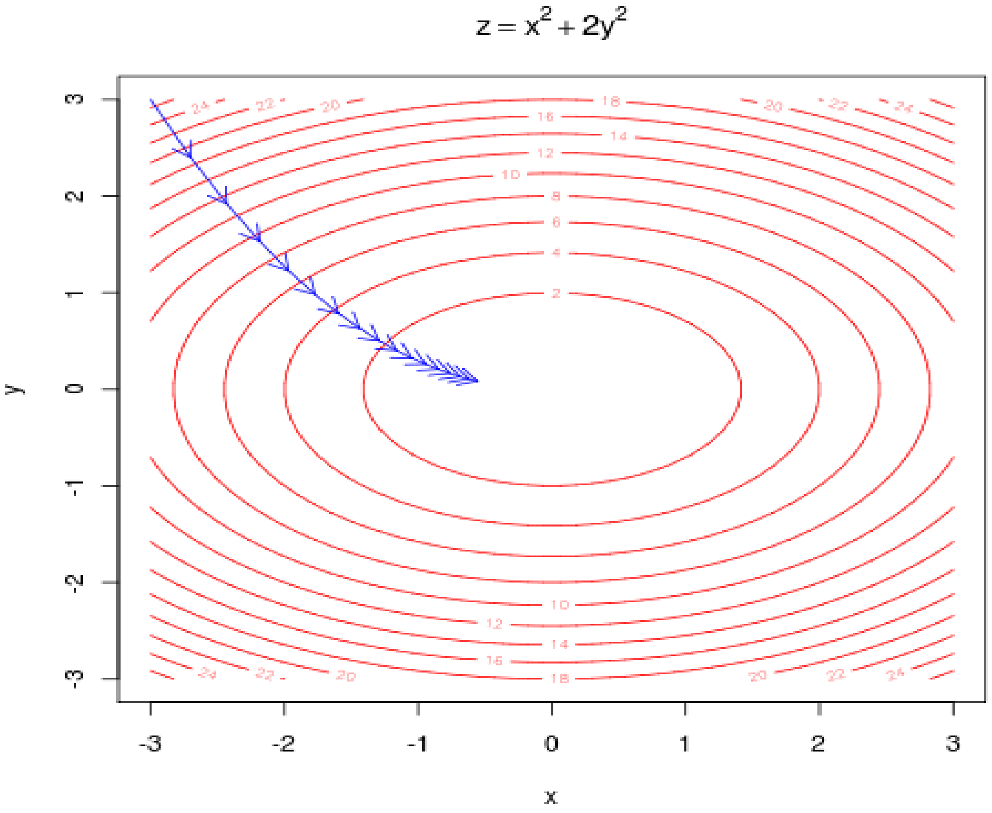


# Digression - Gradient Descent Intuition



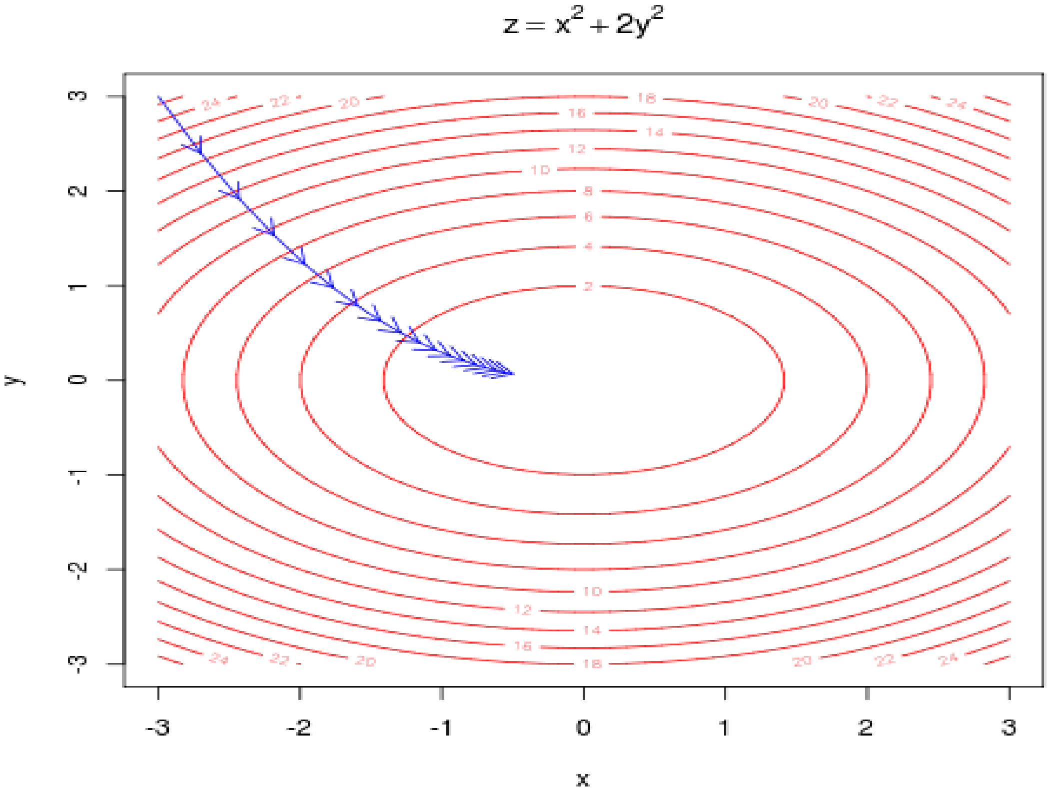

datasciencedojo
unleash the data scientist in you

# Digression - Gradient Descent Intuition



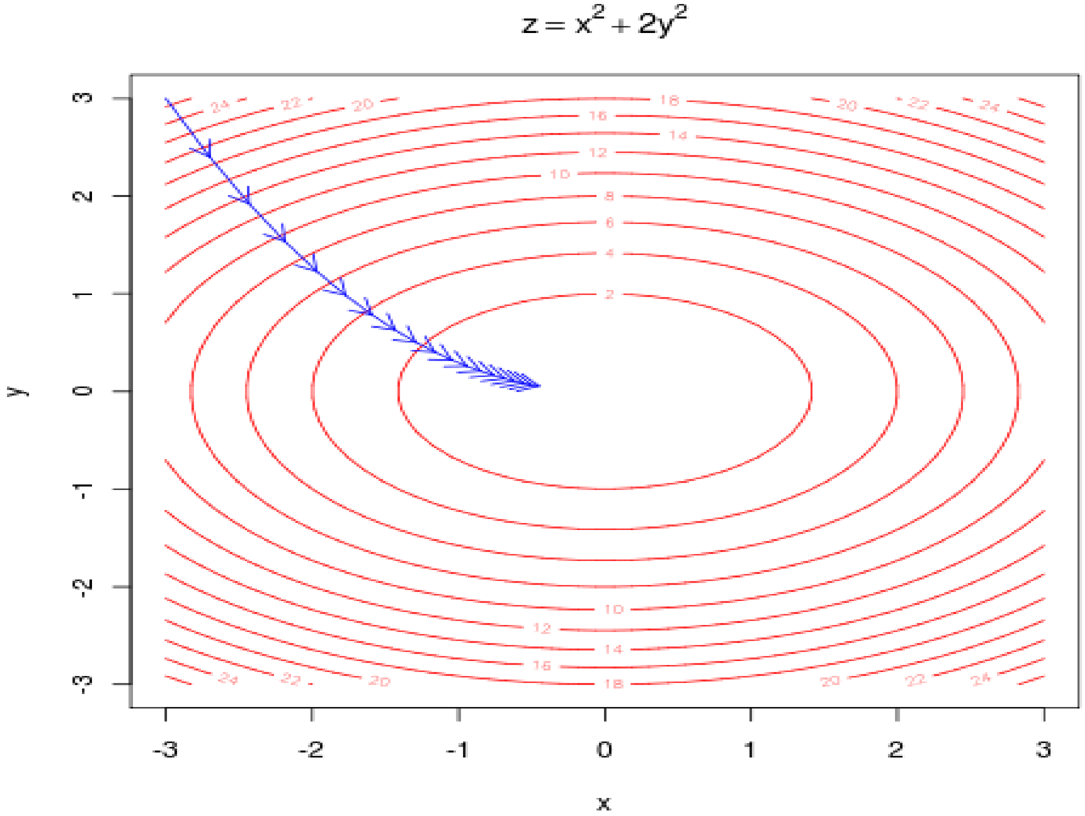

datasciencedojo
unleash the data scientist in you

# Digression - Gradient Descent Intuition

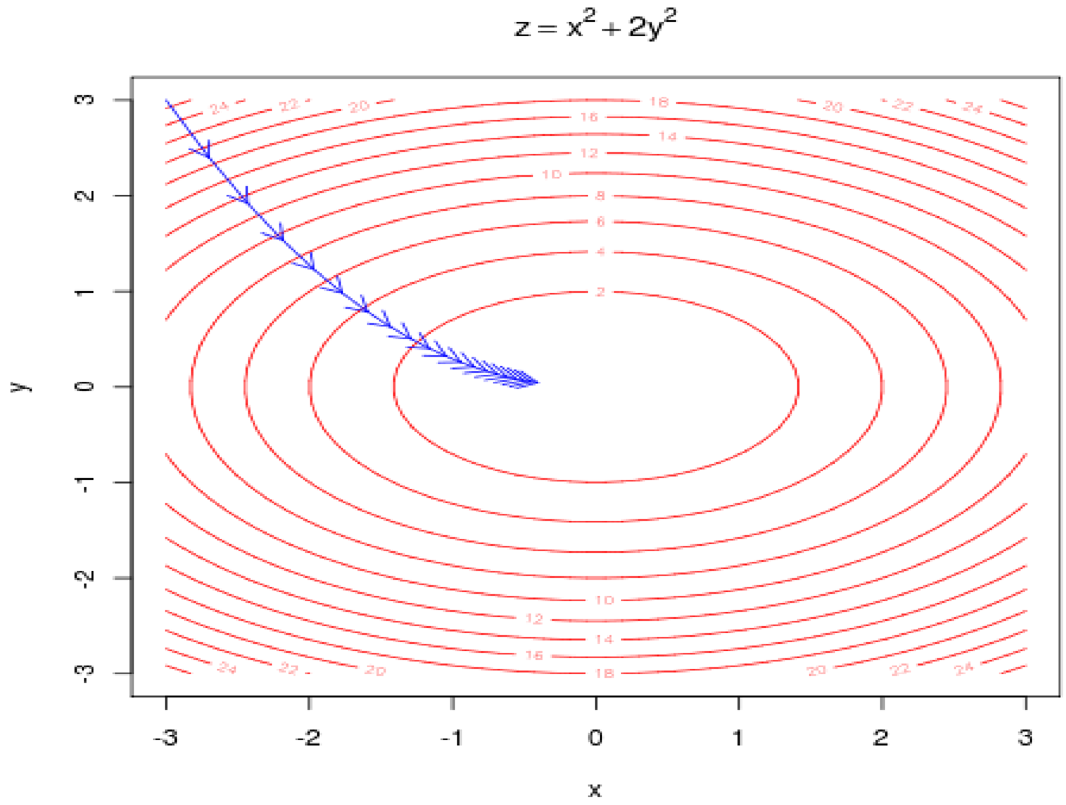



datasciencedojo
unleash the data scientist in you

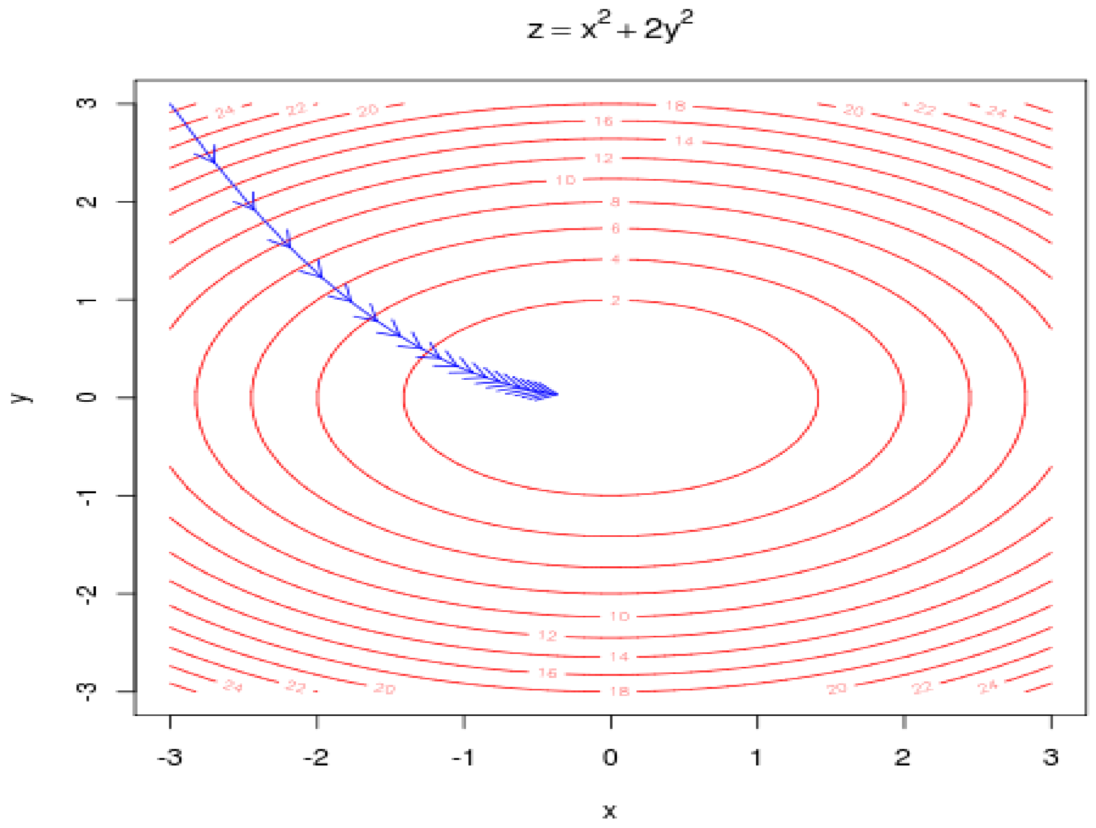
# Digression - Gradient Descent Intuition



# Digression - Gradient Descent Intuition

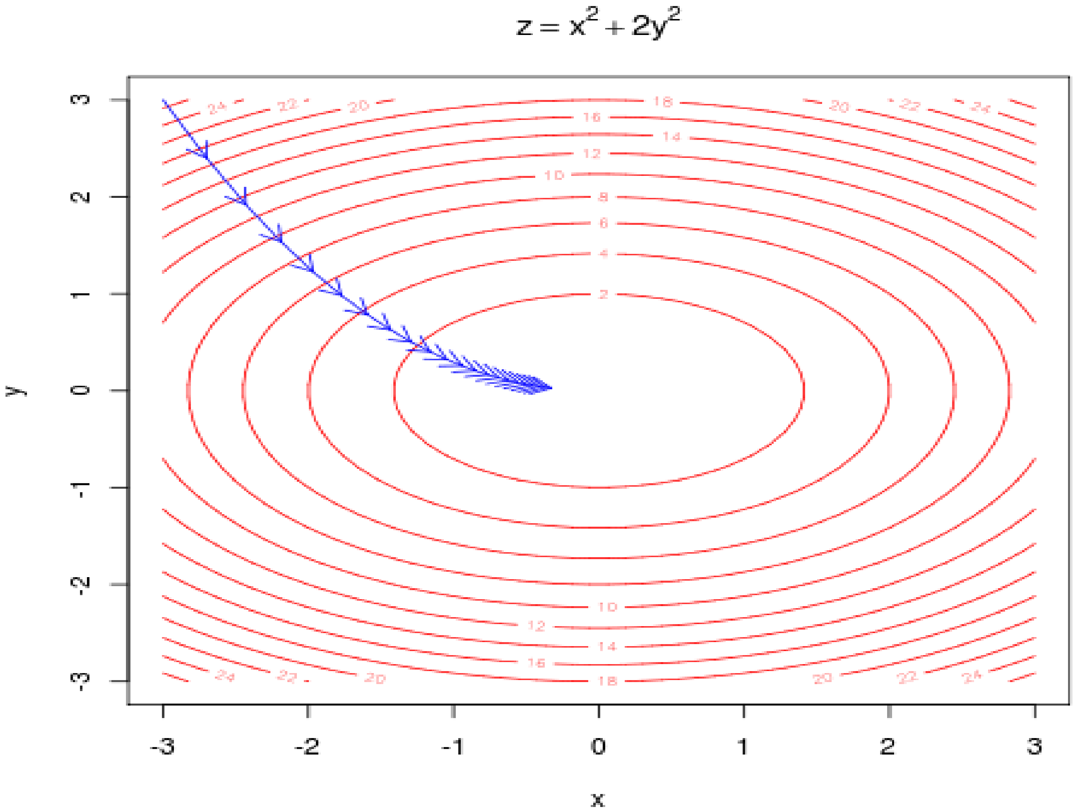


# Digression - Gradient Descent Intuition



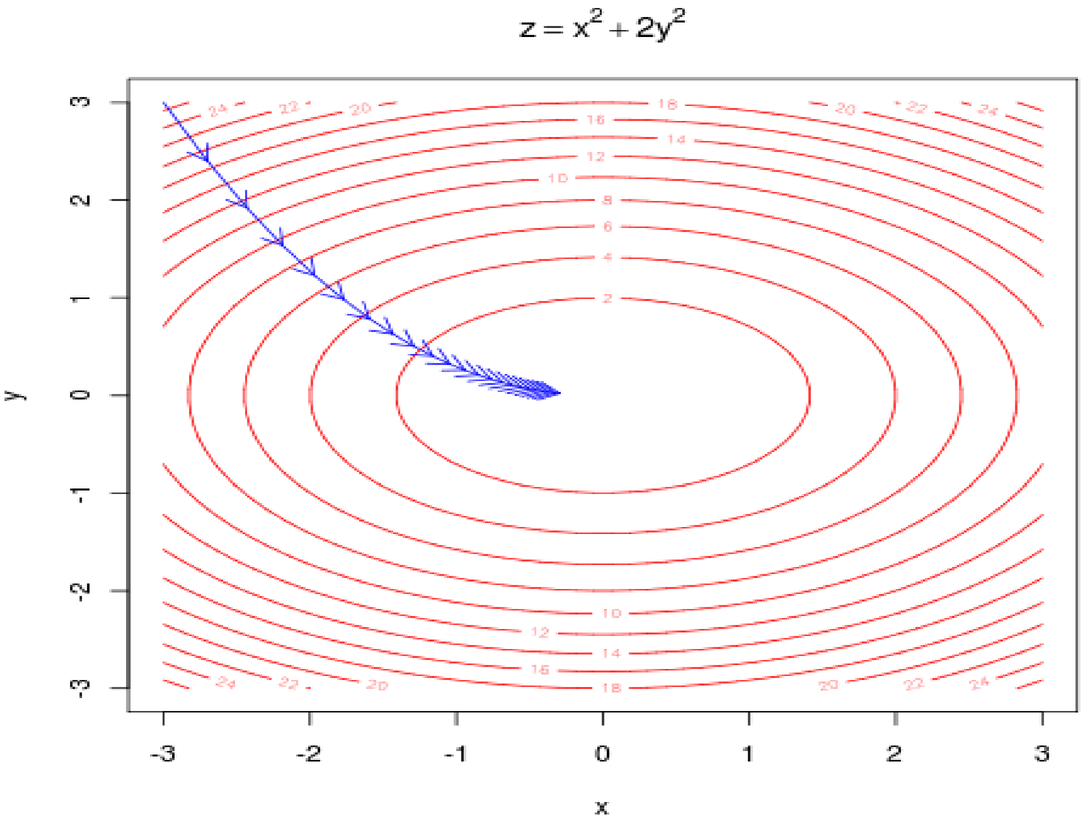

datasciencedojo
unleash the data scientist in you

# Digression - Gradient Descent Intuition

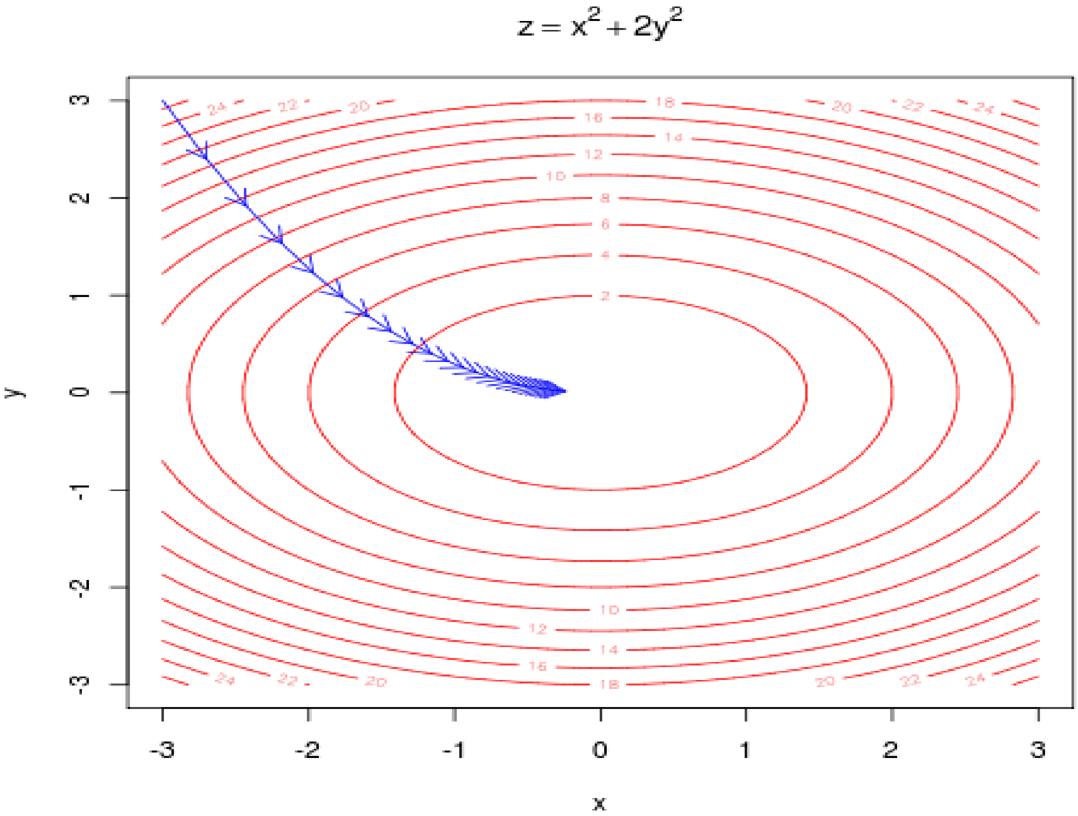



datasciencedojo
unleash the data scientist in you

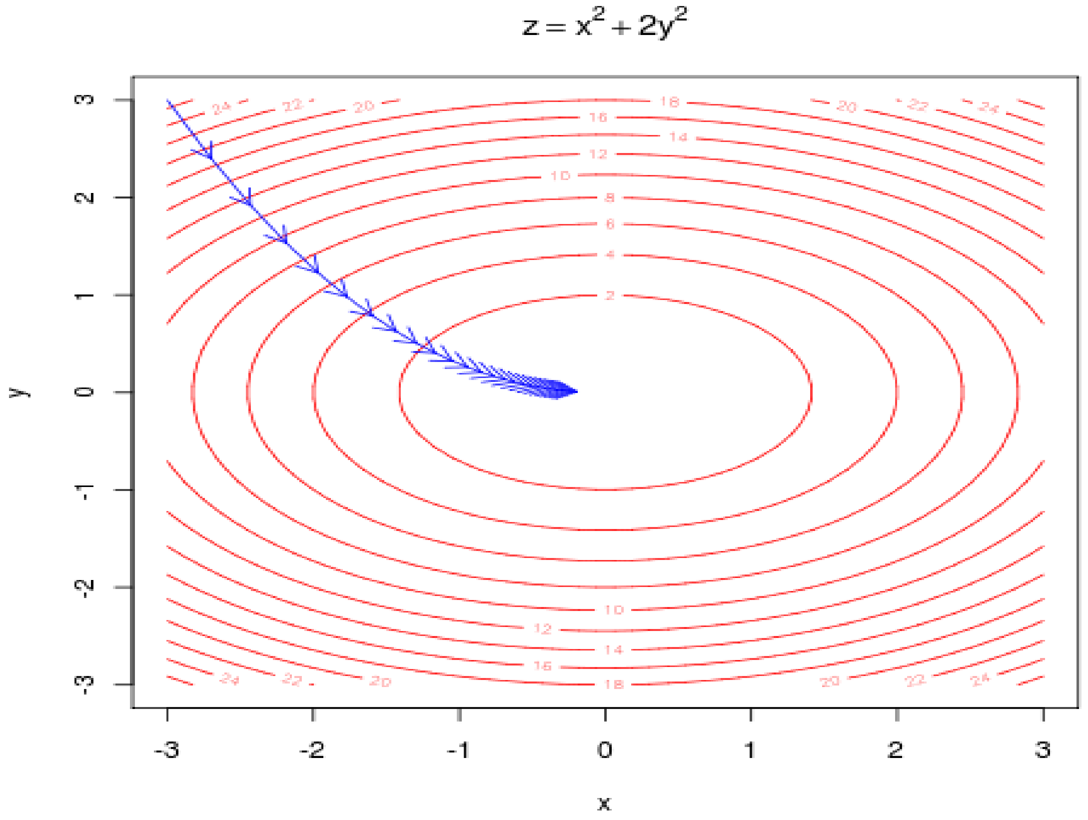
# Digression - Gradient Descent Intuition



# Digression - Gradient Descent Intuition

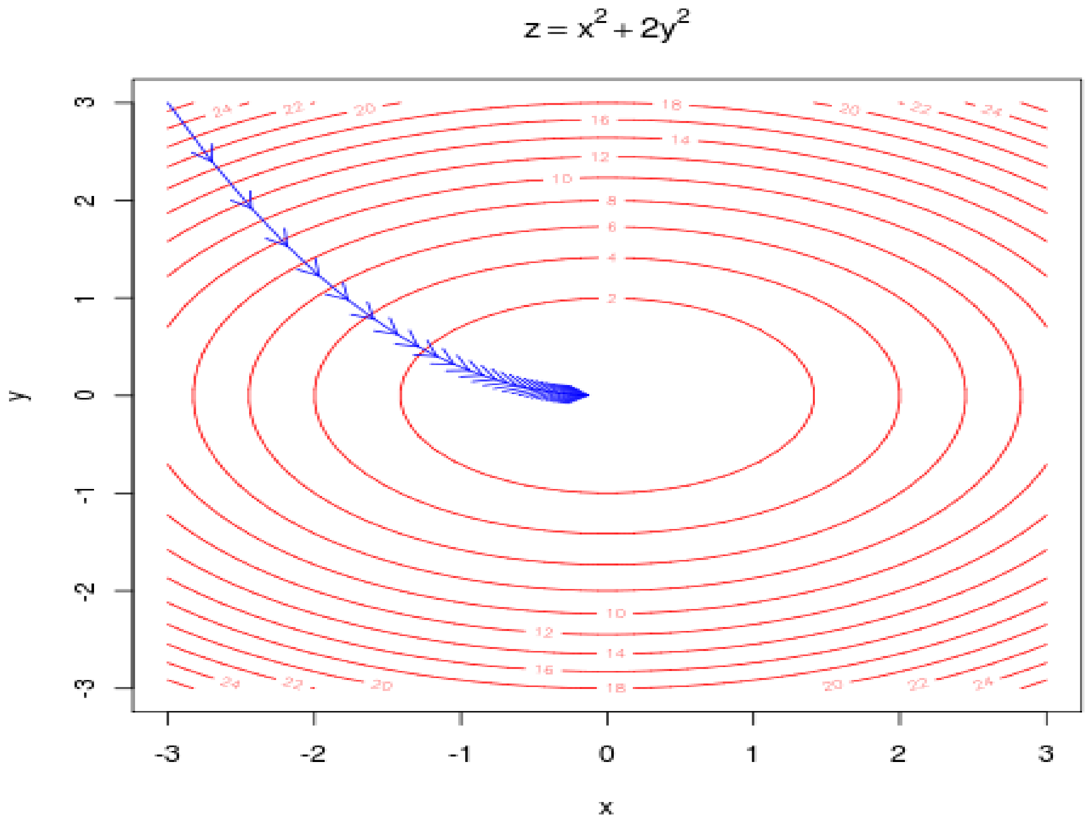


# Digression - Gradient Descent Intuition



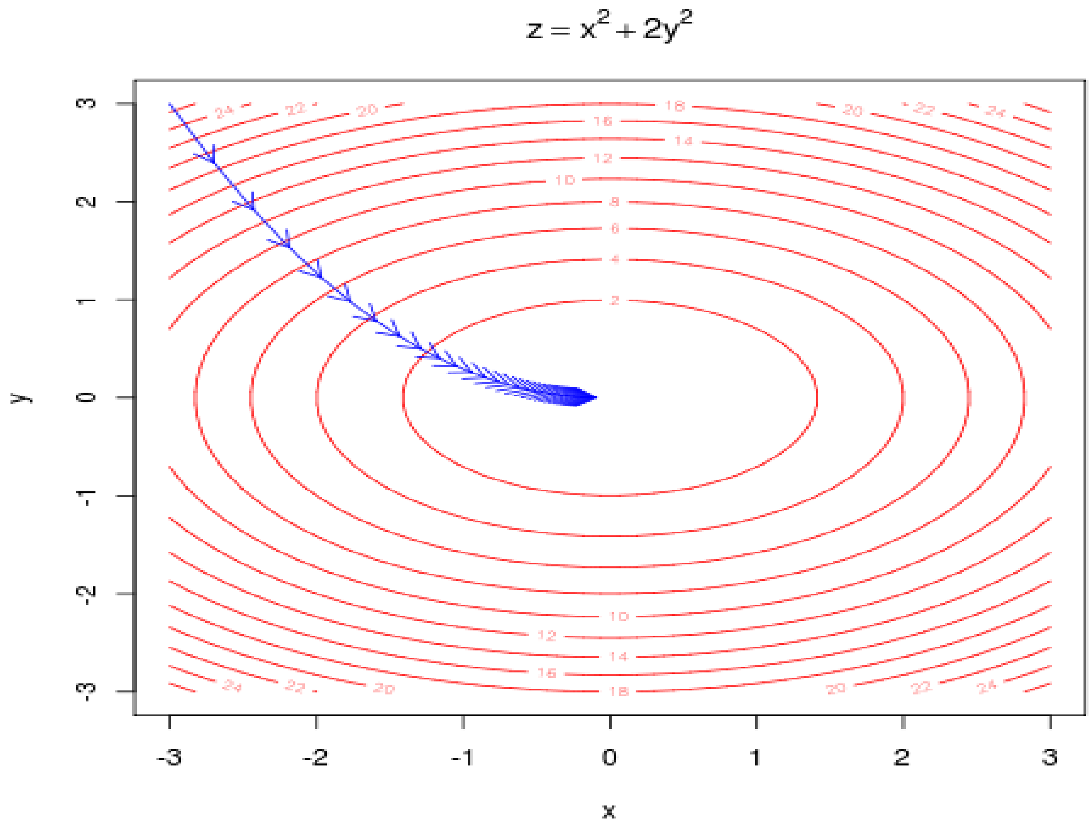

datasciencedojo
unleash the data scientist in you

# Digression - Gradient Descent Intuition



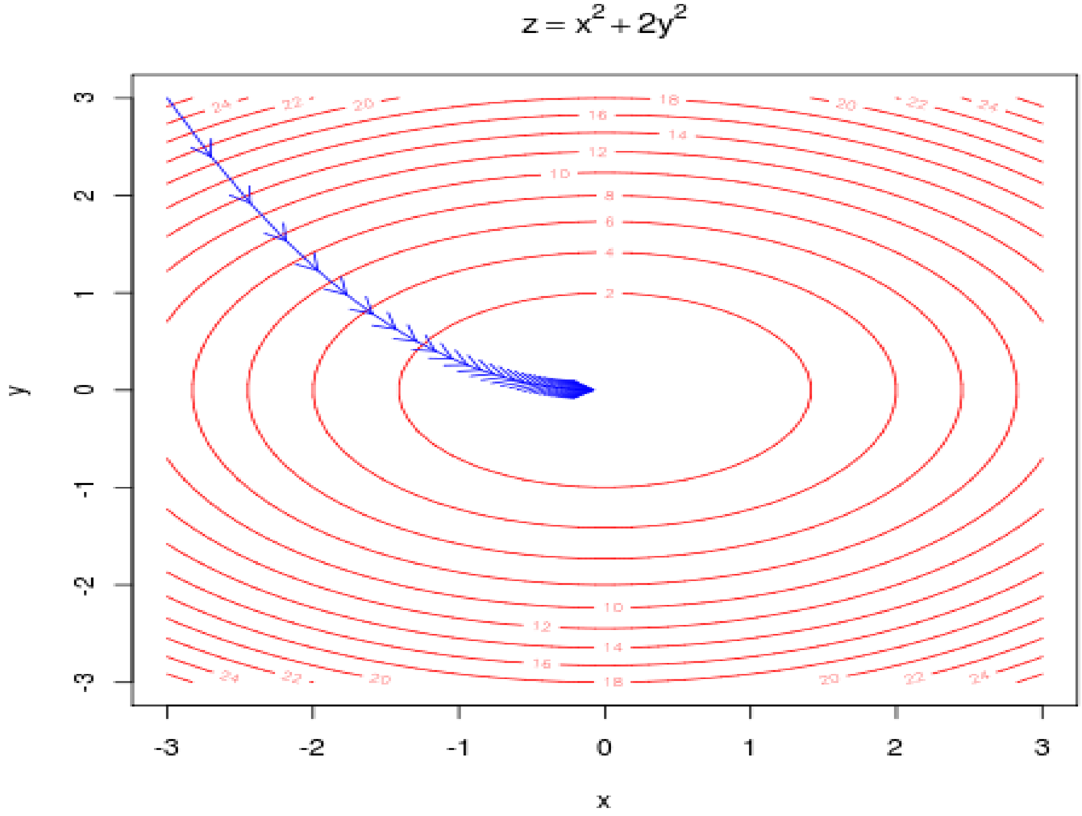

datasciencedojo
unleash the data scientist in you

# Digression - Gradient Descent Intuition



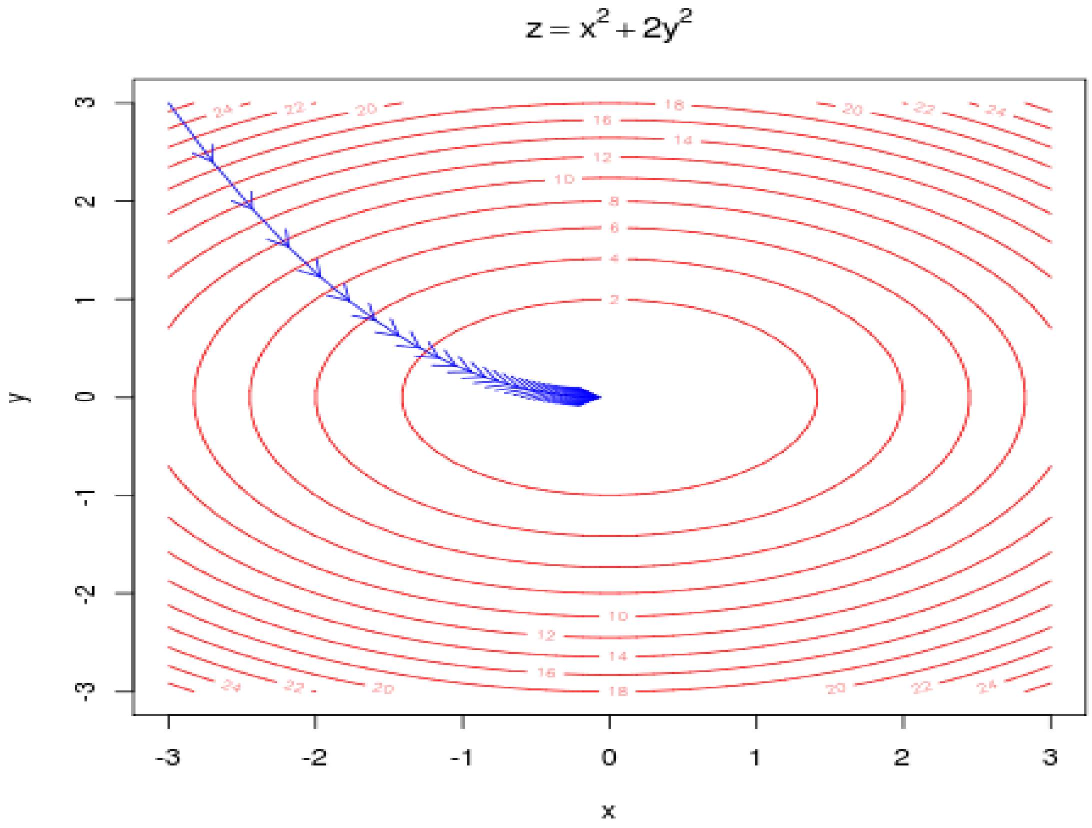

datasciencedojo
unleash the data scientist in you

# Digression - Gradient Descent Intuition




datasciencedojo
unleash the data scientist in you

# Digression - Gradient Descent Intuition




datasciencedojo
unleash the data scientist in you

# Multivariate Linear Regression

# Predict Housing Prices – With Two Features

$x^i{}_1$ $x^i{}_2$ $y^i$

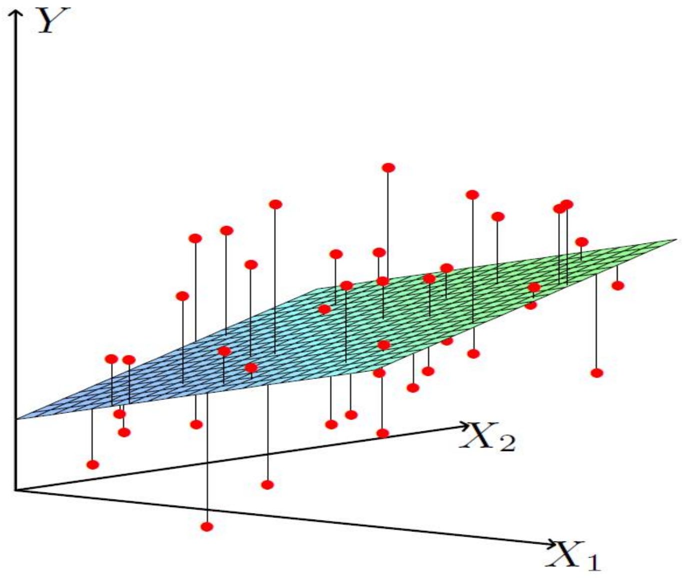
| Living area (feet$^2$) | #bedrooms | Price (1000$s) |
|---|---|---|
| 2104 | 3 | 400 |
| 1600 | 3 | 330 |
| 2400 | 3 | 369 |
| 1416 | 2 | 232 |
| 3000 | 4 | 540 |
| ⋮ | ⋮ | ⋮ |

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

A linear function is just one of the choices to approximate the target variable.

$\theta$ is the space of linear functions mapping the space of input variables to the output/target variables

# Predict Housing Prices – With Two Features




When we have two features, the linearity is in plane instead of line.

# Algebraic Notation

The function approximating the target variable *y* is given by:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

as

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x,$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix}$$

$$\theta^T = [\theta_0 \, \theta_1 \theta_2]$$

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

$$\theta^T x = [\theta_0 \, \theta_1 \theta_2] \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \sum_{i=0}^{n} \theta_i x_i$$

# Batch Gradient Descent

For ONE training examples, we get the following update rule:

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)}.$$

What do we observe here about the magnitude of the update?

For ALL training examples, we get the following update rule:

Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{).}$$

}

# Stochastic Gradient Descent

- Consider the following algorithm:

Loop {

for i=1 to m, {

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{)}.$$

}

}

- This algorithm updates the parameters $\theta_j$ using each training example instead of all training examples.
- If the training set is big i.e., m is large, this technique converges quicker than batch gradient descent.
- Stochastic gradient descent may oscillate around the minimum of $J(\theta)$ and may not completely converge

datasciencedojo
unleash the data scientist in you

# Batch vs. Stochastic Gradient Descent

## Batch Gradient Descent

Repeat until convergence {

$$\theta_j := \theta_j + \alpha \sum_{i=1}^{m} \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{).}$$

}

- To update each parameter value, scan through the whole training data
- Converges to the minimum value slowly
- Preferred for small datasets

## Stochastic Gradient Descent

Loop {

for i=1 to m, {

$$\theta_j := \theta_j + \alpha \left(y^{(i)} - h_\theta(x^{(i)})\right) x_j^{(i)} \qquad \text{(for every } j\text{).}$$

}

}

- Update the parameter values with one training example at a time
- Converges to the ‘proximity’ of minimum value fast but may keep oscillating near the minimum
- Preferred for large datasets

datasciencedojo
unleash the data scientist in you

# Probabilistic Interpretation of Linear Regression

- We will discuss why the least-squares cost function $J(\theta)$ is a reasonable choice.
- Assume that the inputs and the target variable are related by the following equation:

$$y^{(i)} = \theta^T x^{(i)} + \epsilon^{(i)}$$

datasciencedojo
unleash the data scientist in you

# Probabilistic Interpretation of Linear Regression

- Where the $\epsilon^i$ term captures the un-modeled effects:
  - Random noise
  - Some relevant feature not taken into consideration
- Assumptions:
  - $\epsilon^i$ is IID (Independently and Identically Distributed)
  - Normal distribution with mean 0 and variance $\sigma^2$
  - variance $\sigma^2$ is random

$$\epsilon^i = N(0, \sigma^2)$$

# Probabilistic Interpretation of Linear Regression

- The density of $\epsilon^i$ is given by $p(\epsilon^{(i)}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(\epsilon^{(i)})^2}{2\sigma^2}\right)$

We can rewrite:

$$p(y^{(i)}|x^{(i)};\theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right)$$

This is the distribution of $y^i$ given $x^i$ and parameterized by $\theta$

$$y^{(i)} \mid x^{(i)};\theta \sim \mathcal{N}(\theta^T x^{(i)}, \sigma^2)$$

# Probabilistic Interpretation of Linear Regression

- Probability of observing the data as a function of $\theta$ is given by:

$$L(\theta) = L(\theta; X, \vec{y}) = p(\vec{y}|X; \theta).$$

- This is known as the likelihood function
- Due to the independence assumption, we can rewrite:

$$\begin{aligned} L(\theta) &= \prod_{i=1}^{m} p(y^{(i)} \mid x^{(i)}; \theta) \\ &= \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right). \end{aligned}$$

# Maximizing The Likelihood Function

- Principle of maximum likelihood: Choose $\theta$ that makes the observed data as high probability as possible. In other words, choose $\theta$ that maximizes $L(\theta)$.
- For mathematical ease, we can maximize the log of the likelihood function $L(\theta)$ instead.

# Maximizing The Log Likelihood Function

$$\begin{aligned}
\ell(\theta) &= \log L(\theta) \\
&= \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= \sum_{i=1}^{m} \log \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(y^{(i)} - \theta^T x^{(i)})^2}{2\sigma^2}\right) \\
&= m \log \frac{1}{\sqrt{2\pi}\sigma} - \frac{1}{\sigma^2} \cdot \frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2.
\end{aligned}$$

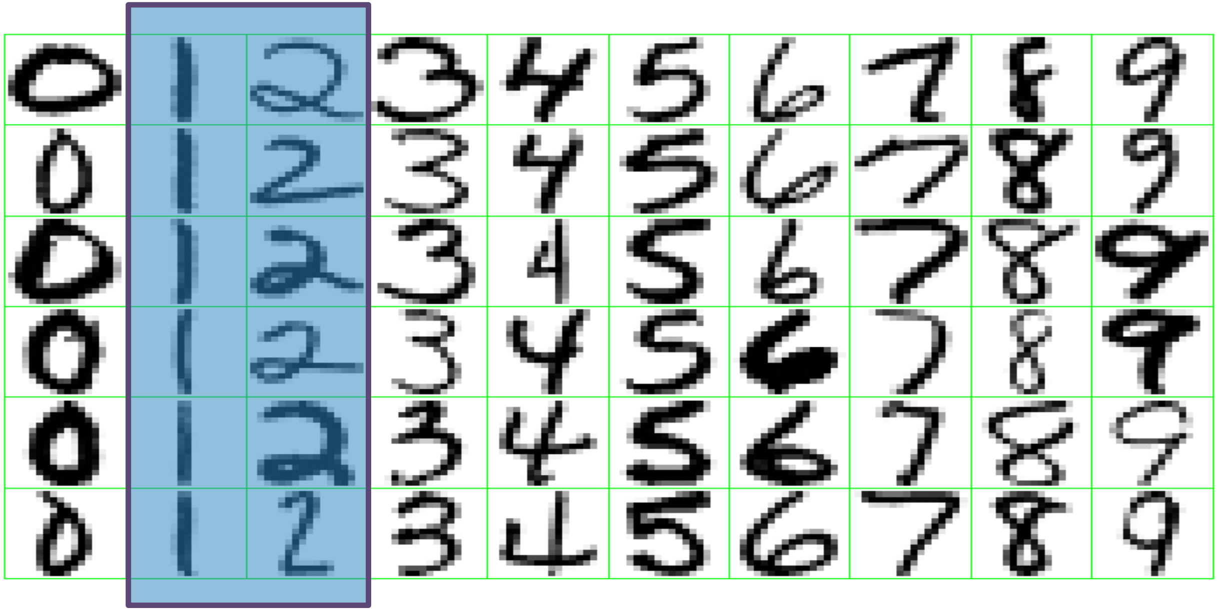
$$\frac{1}{2} \sum_{i=1}^{m} (y^{(i)} - \theta^T x^{(i)})^2$$

Maximizing $L(\theta)$ is equivalent to minimizing $J(\theta)$

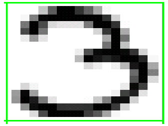
# Linear Regression Using R

- We will learn how to use linear regression for handwritten digit recognition now
- Distinguishing between 2s and 3s
- We will see some R code
  - For processing the inputs
  - Applying linear regression to learn a hypothesis
- You can download R by searching ‘R download’

# Example: Handwritten Digit Recognition



# Extracting Features For Learning



$\{x_1, x_2, x_3 \ldots\ldots\ldots\ X_{256}, y = \text{'three'}\}$

- Each xi corresponds to a feature value in the image
- y is a label of the training data; can be numeric or categorical, '3' or 'three'
- Each image is converted to row vectors and the appropriate learning algorithm is used
- Convention
  - $x_i$ represents the ith feature in a training sample
  - y represents the label for the training sample

# QUESTIONS

datasciencedojo
unleash the data scientist in you