Numerical Optimization for ML&DL (NOFML&DL)

Information Technology Institute

Session 2

Review: Differentiation (derivative of a function).

Review: Gradient Descent (GD) Algorithm

Gradient of Multivariable Function.

GD Applied to Single Variable Linear Regression (LR).

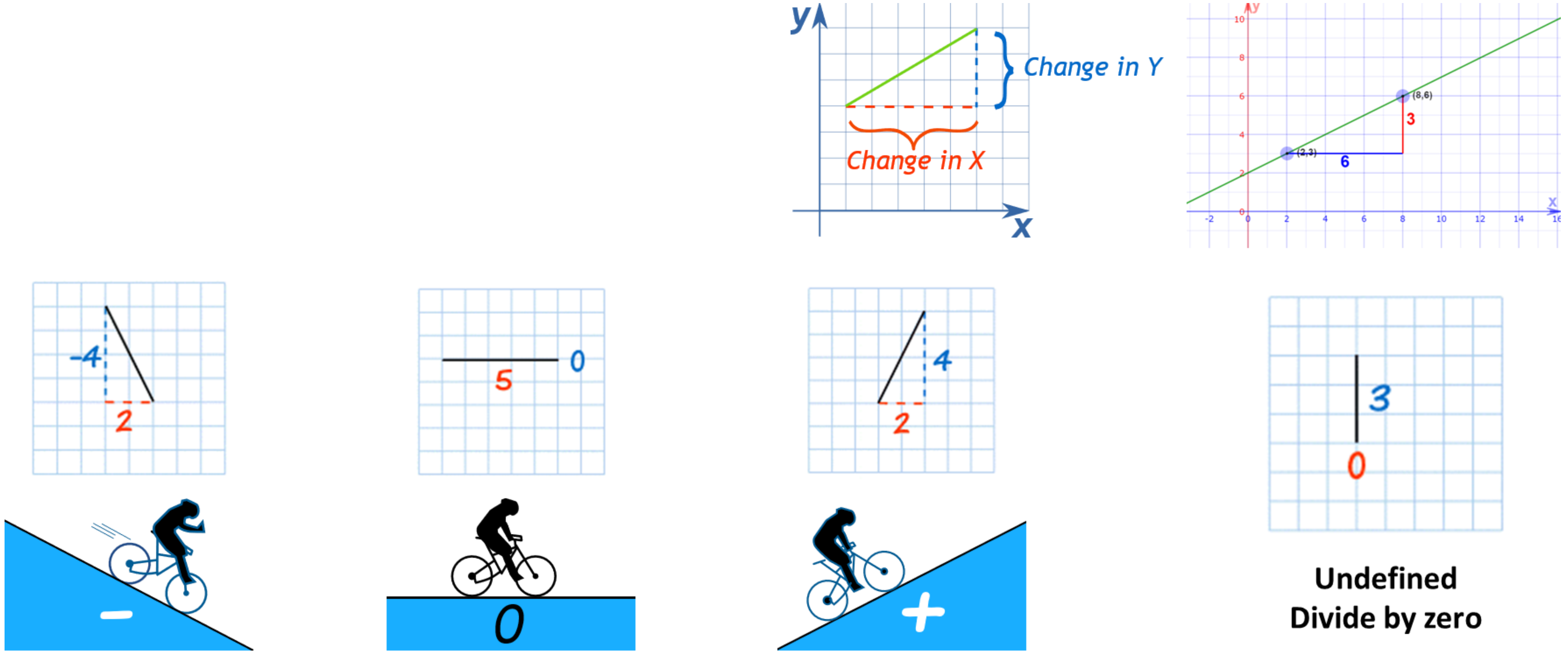
GD Applied to Multivariable LR.

Local vs. Global Minimum.

Contour Plot.

Features Scaling.

Review of differentiation and the concept of gradient

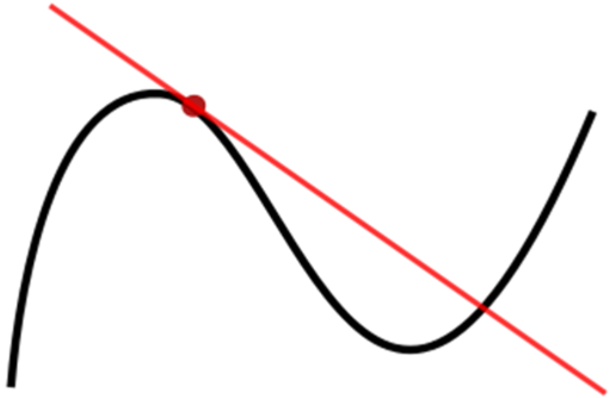


Gradient

- **What is the gradient?**
- **Gradient is the slope**
- $$\boldsymbol{Gradient = \frac{Change\ in\ Y}{Change\ in\ X} = \frac{Rise}{Run}}$$

Gradient

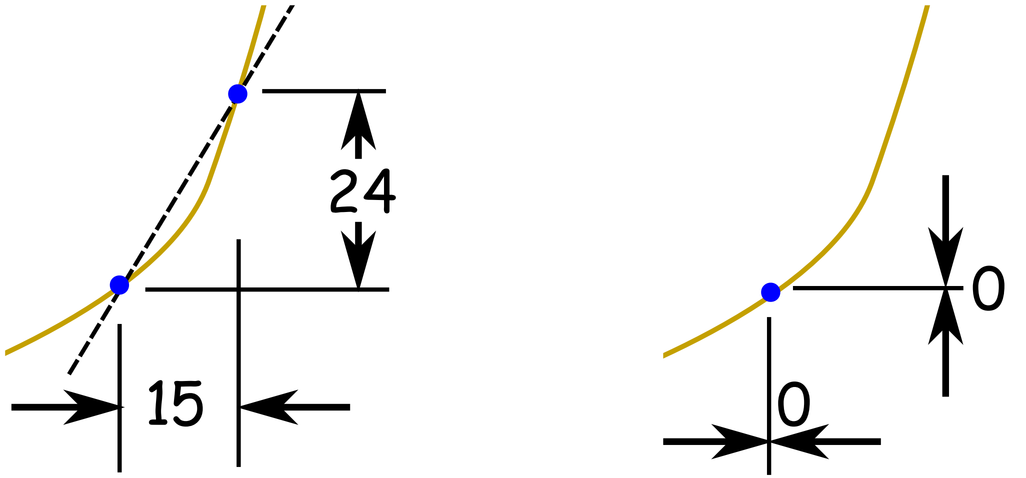
- The **derivative** of a function of a real variable measures the sensitivity to change of the function value (output value) with respect to a change in its argument (input value).
- For example, the derivative of the position of a moving object with respect to time is the object's velocity: this measures how quickly the position of the object changes when time advances.



The graph of a function, drawn in black, and a tangent line to that function, drawn in red. The slope of the tangent line is equal to the derivative of the function at the marked point.

Gradient

- We can find an average slope between two points.
- But how do we find the slope at a point?



$$\text{average slope} = \frac{24}{15}$$

$$\text{slope} = \frac{0}{0} = ???$$

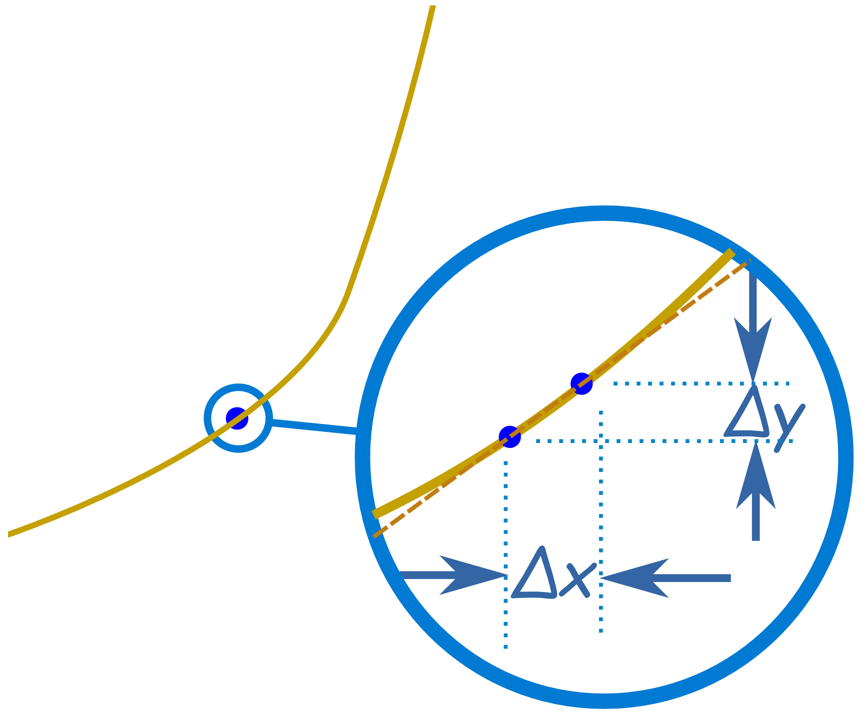
Gradient

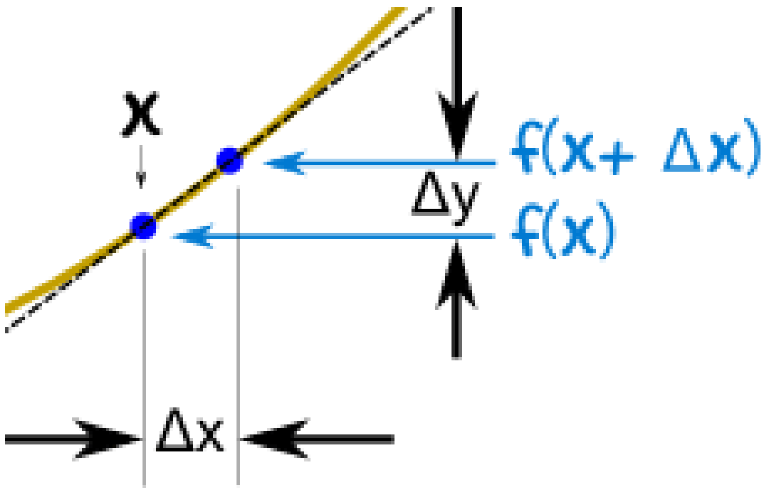
- But with derivatives we use a small difference then have it shrink towards zero

- $\frac{\Delta y}{\Delta x} = \frac{f(x+\Delta x) - f(x)}{\Delta x}$
- **Gradient at x = derivative at x**

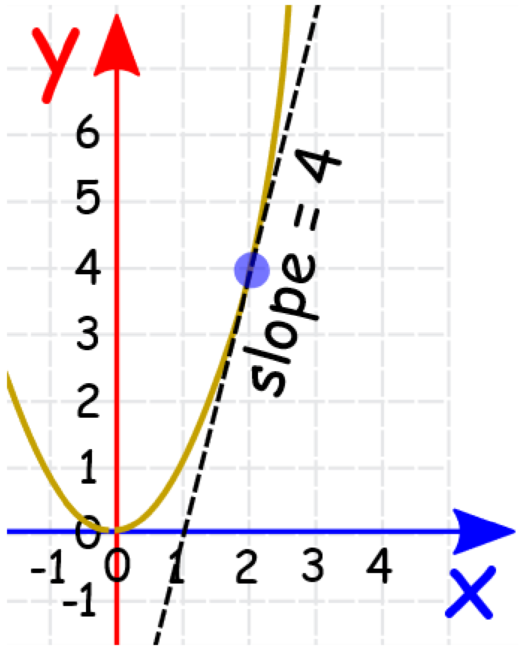
=

$$\lim_{\Delta x \to 0} \left(\frac{f(x+\Delta x) - f(x)}{\Delta x} \right)$$

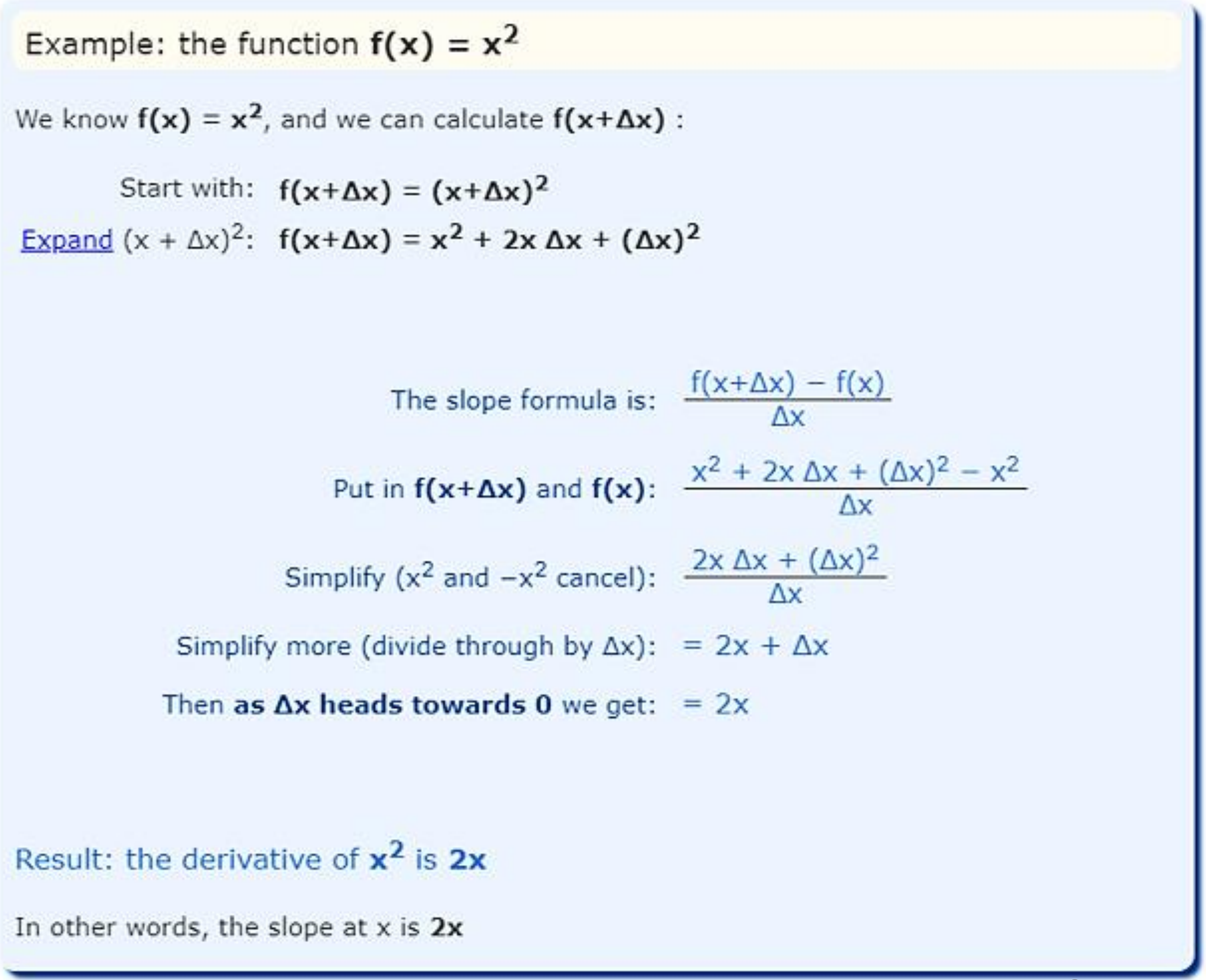




Gradient



$$f'(x) = \frac{d}{dx}(x^2) = 2x$$



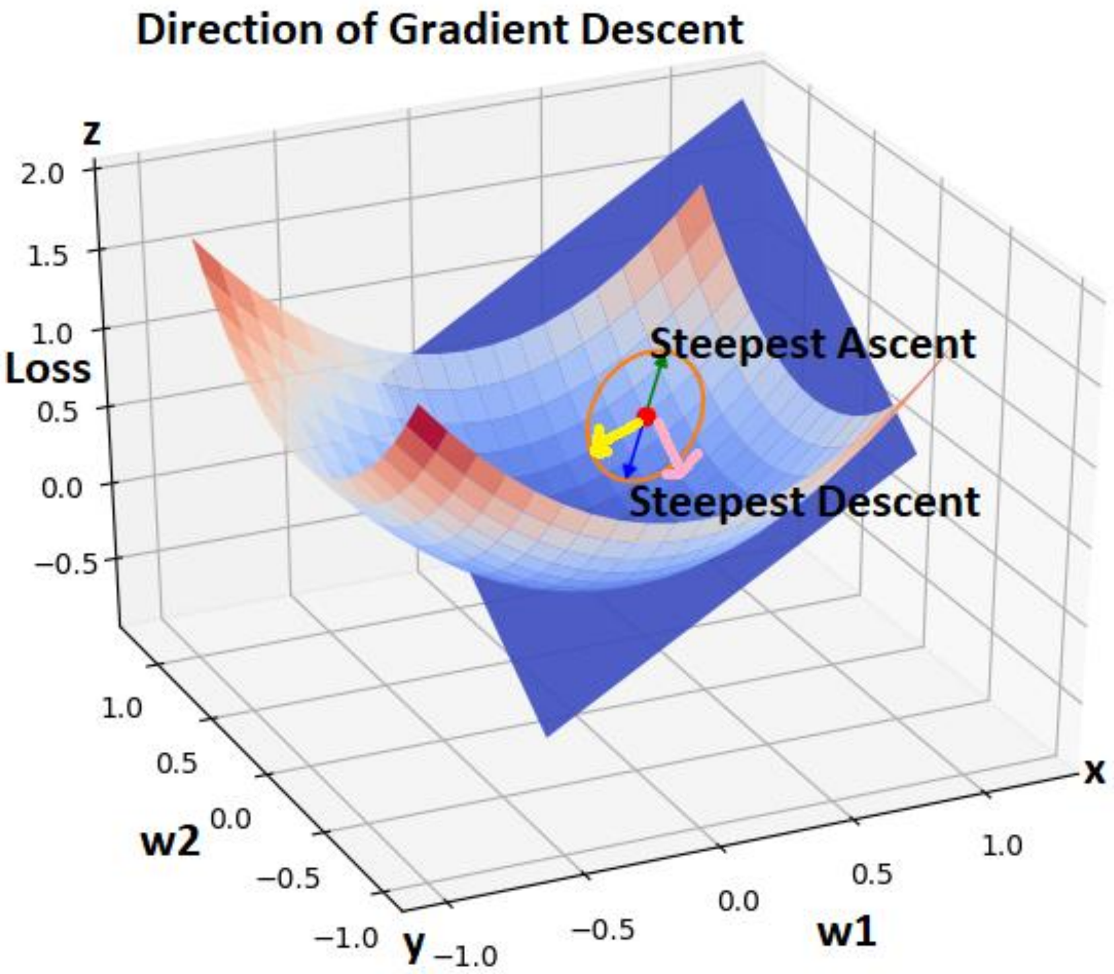
Gradient of a Multivariable function

Gradient of a Multivariable Function

- Gradient is a vector;

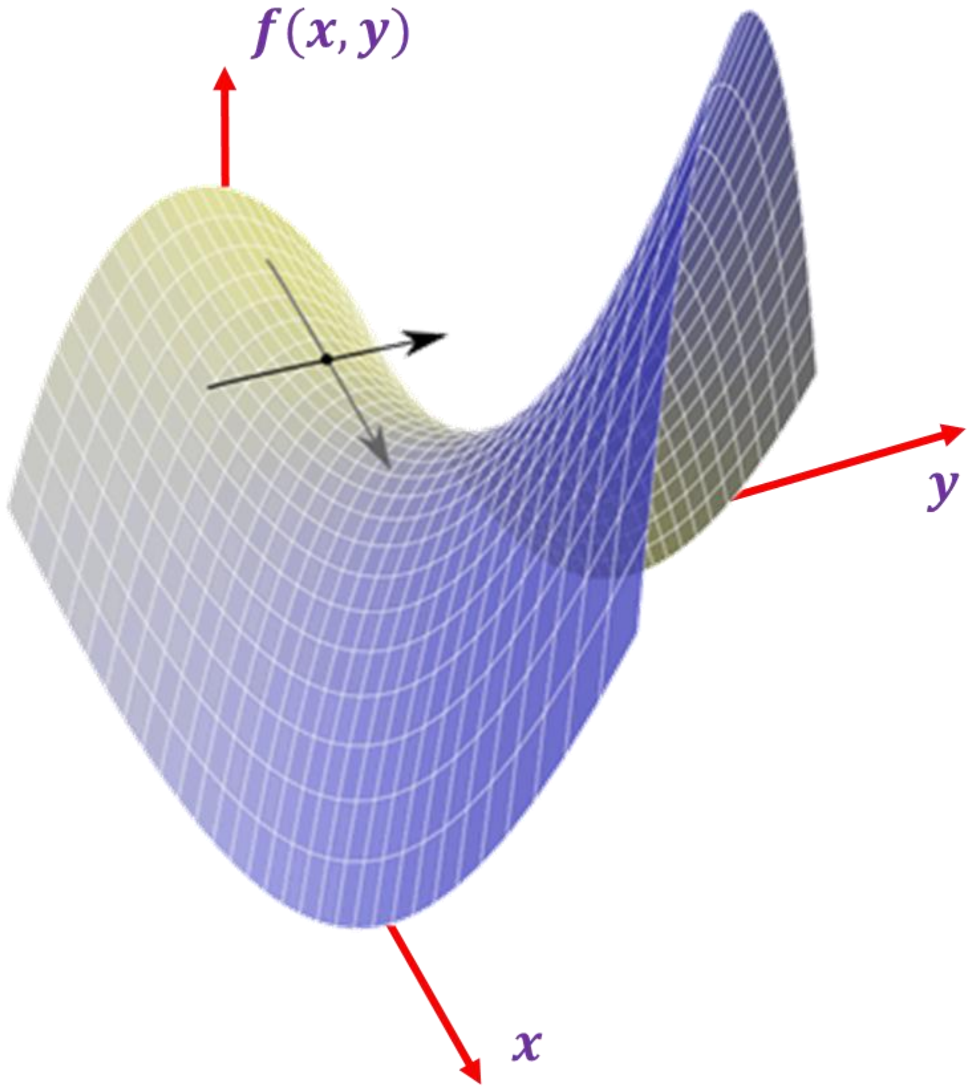
- $\boldsymbol{\nabla J} = \begin{bmatrix} \frac{\partial J}{\partial \theta_0} \\ \frac{\partial J}{\partial \theta_1} \end{bmatrix}$

(It is a special case of the Jacobian matrix; therefore, it is sometimes called Jacobian vector)



Gradient of a Multivariable Function

- A **Partial Derivative** is the rate of change of a multi-variable function when all, but one variable is held fixed.
- **Example:**
  - a function for a surface that depends on two variables x and y.
  - When we find the slope in the x direction (while keeping y fixed) we have found a partial derivative.

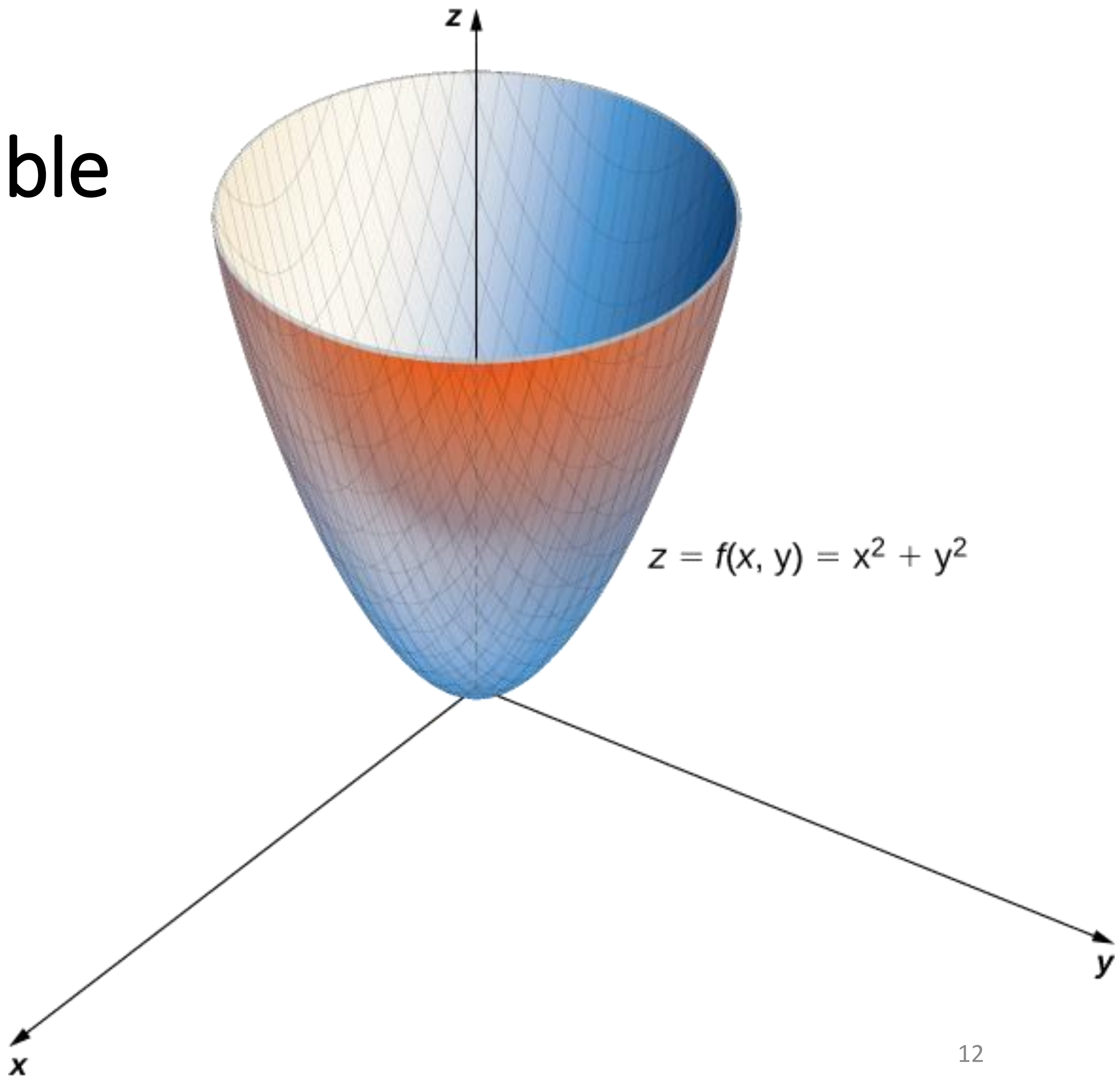


Gradient of a Multivariable Function

- **Example;**

- $\frac{\partial f}{\partial x} = 2x \ ; \frac{\partial f}{\partial y} = 2y$

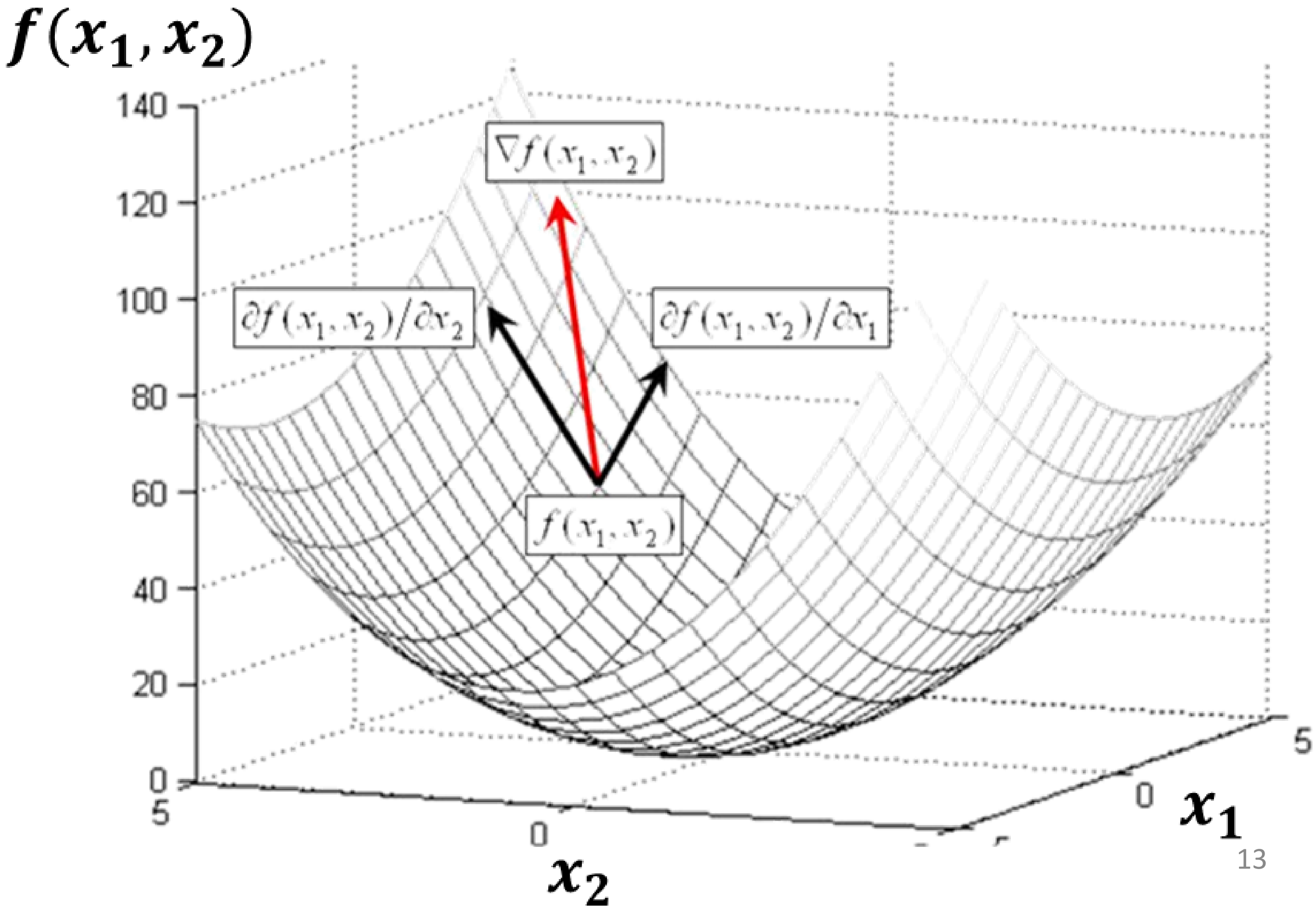
- $Gradient \ = \nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} 2x \\ 2y \end{bmatrix}$



Gradient of Multivariable Function

- Gradient is a vector that points to the opposite of steepest descent direction.
- Can be generalized to **n** dimensions
- ***p*** is any point at where the gradient is calculated.

Gradient vector points in direction of steepest ascent of function



Linear Regression & Loss Function

Implementation Steps of GD Applied to Linear Regression (Single Variable)

- **Step 1:** Initialize parameters **(θ_0 & θ_1)** with random value or simply zero. Also choose the **Learning rate**.
- **Step 2:** Use (θ_0 & θ_1) to predict the output **$h_\theta(x) = \theta_0 + \theta_1 x$**.
- **Step 3:** Calculate the cost function $\boldsymbol{J}(\theta_0, \theta_1)$.
- **Step 4:** Calculate the gradient.
- **Step 5:** Update the parameters (simultaneously).
- **Step 6:** Repeat from 2 to 5 until converge to the minimum or achieve maximum iterations.

Linear Regression with Mean Squared Error Cost Function

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

Parameters: θ_0, θ_1

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

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

GD Applied to Linear Regression (Single Variable)

Gradient descent algorithm

$$
\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad \theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \\
&\qquad \text{(for } j = 1 \text{ and } j = 0) \\
&\}
\end{aligned}
$$

Linear Regression Model

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

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

GD Applied to Linear Regression (Single Variable)

$$
\begin{aligned}
&\text{repeat until convergence } \{ \\
&\quad \theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \\
&\quad \theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) \cdot x^{(i)} \\
&\}
\end{aligned}
$$

GD Algorithm Implementation Notes

- Parameters should be updated simultaneously.
- Learning step will decrease as you become closer to the minimum. Even with fixed learning rate.
- Do not use very large learning rate in order not to overshoot.
- Do not use very small learning rate in order not to go very slowly.

GD applied to Multivariable Linear Regression

GD Applied to Multivariable LR

- In single variable LR;

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x$$

- In multivariable LR;

$$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$$

- Note that $x_0 = 1$
- The hypothesis can be expressed as;

$$h_\theta(x) = \Theta \cdot X = \Theta^T X$$

x_0	x_1	x_2	x_3	x_4	y
Bias (intersect) multiplier	**Area**	**House Age**	**Number of Rooms**	**Number of Bedrooms**	**Price (e+06)**
1	79545	5	7	4	1.059
1	79248	6	6	3	1.505
1	61287	5	8	5	1.058
1	63345	7	5	3	1.260
1	59982	5	7	4	6.309

GD Applied to Multivariable LR

Hypothesis: $h_\theta(x) = \theta^T x = \theta_0 x_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$

Parameters: $\theta_0, \theta_1, \ldots, \theta_n$

Cost function:

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

Gradient descent:

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \ldots, \theta_n)$$

} (simultaneously update for every $j = 0, \ldots, n$)

GD Applied to Multivariable LR

Gradient Descent

Previously (n=1):

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha \underbrace{\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})}_{\frac{\partial}{\partial \theta_0}J(\theta)}$$

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

(simultaneously update θ_0, θ_1)

$\}$

New algorithm $(n \geq 1)$:

Repeat $\{$

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

(simultaneously update θ_j for $j = 0, \ldots, n$)

$\}$

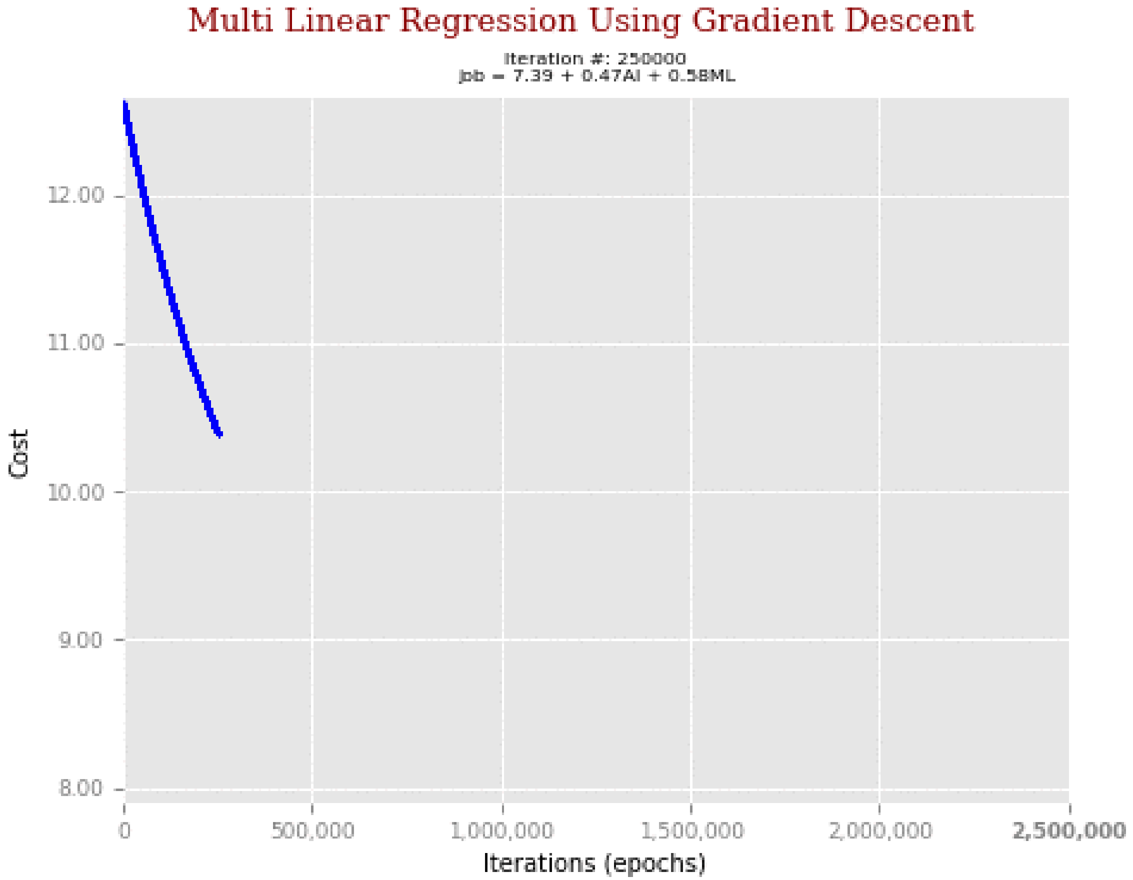
$$\theta_0 := \theta_0 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

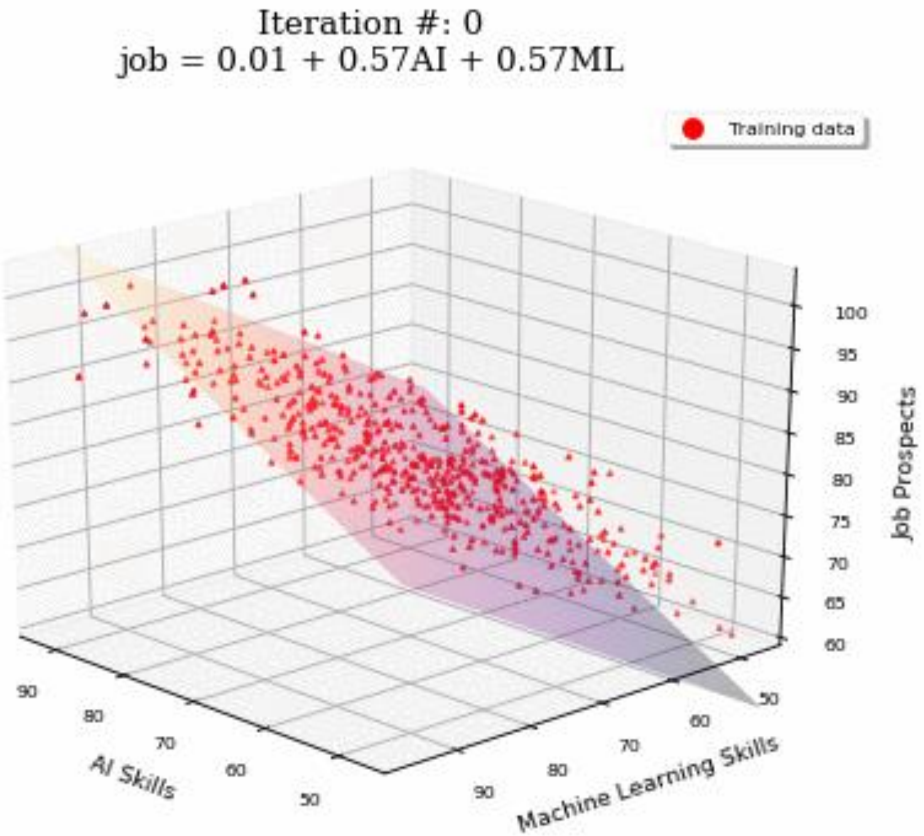
$$\theta_1 := \theta_1 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_1^{(i)}$$

$$\theta_2 := \theta_2 - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_2^{(i)}$$

$\ldots$

GD Applied to Multivariable Linear Regression



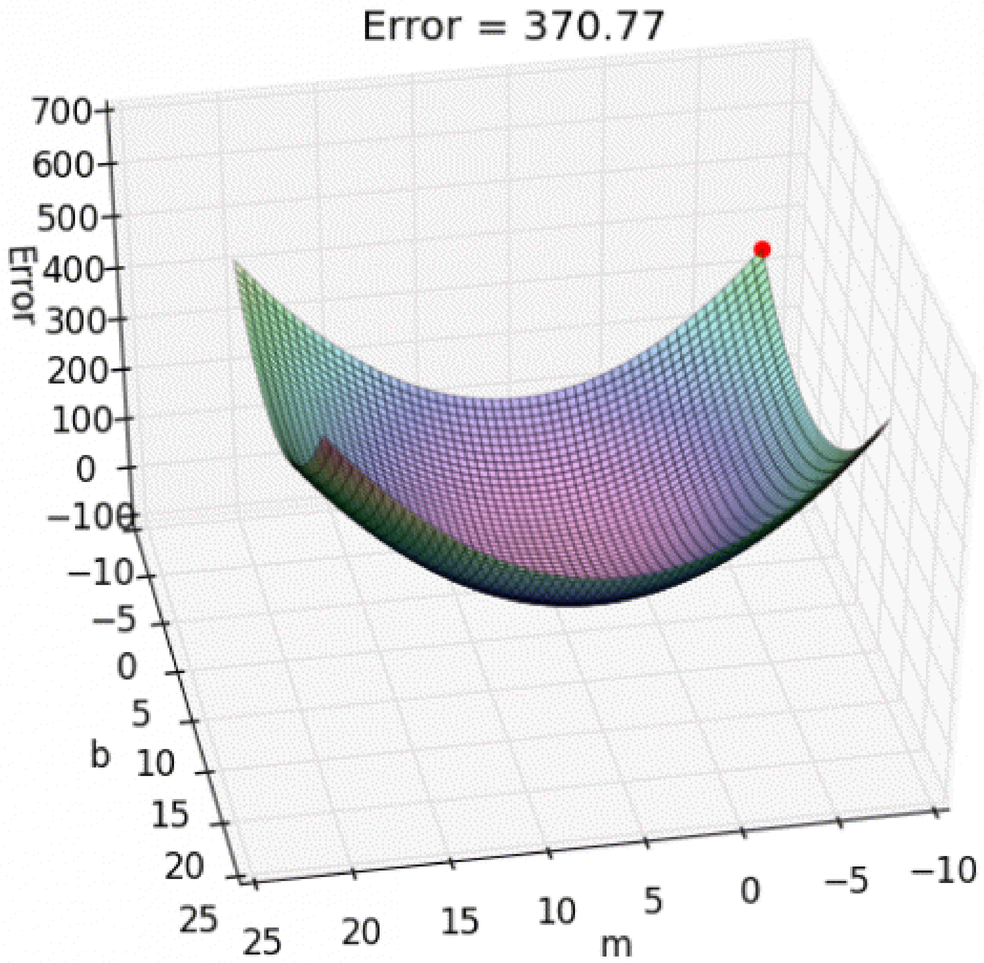


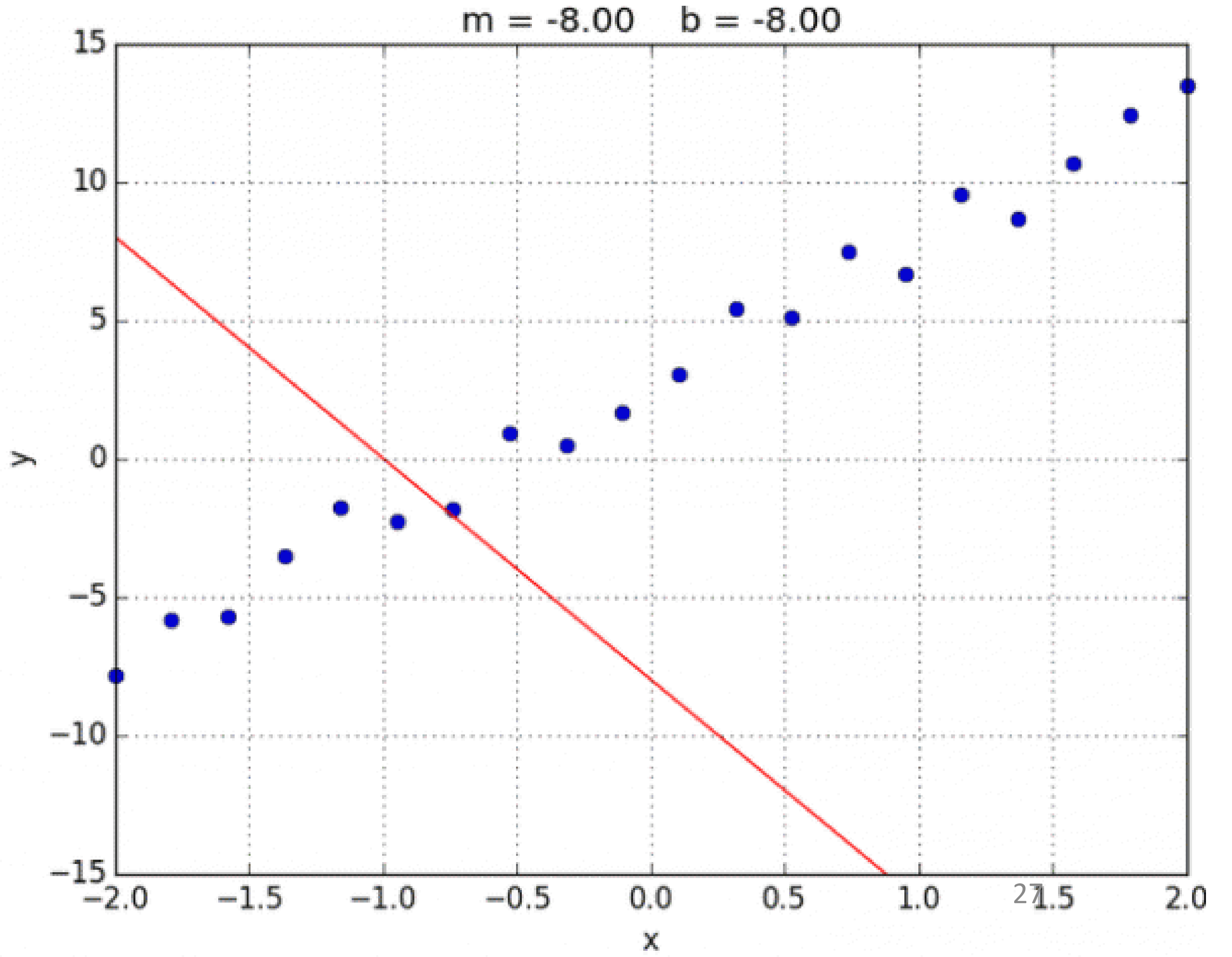
Batch/Vanilla GD

- **The main advantages:**
  - We can use fixed learning rate during training without worrying about learning rate decay.
  - It has straight trajectory towards the minimum and it is guaranteed to converge in theory to the global minimum if the loss function is convex and to a local minimum if the loss function is not convex.
  - It has unbiased estimate of gradients. The more the examples, the lower the standard error.
- **The main disadvantages:**
  - Even though we can use vectorized implementation, it may still be slow to go over all examples especially when we have large datasets.
  - Each step of learning happens after going over all examples where some examples may be redundant and don’t contribute much to the update.

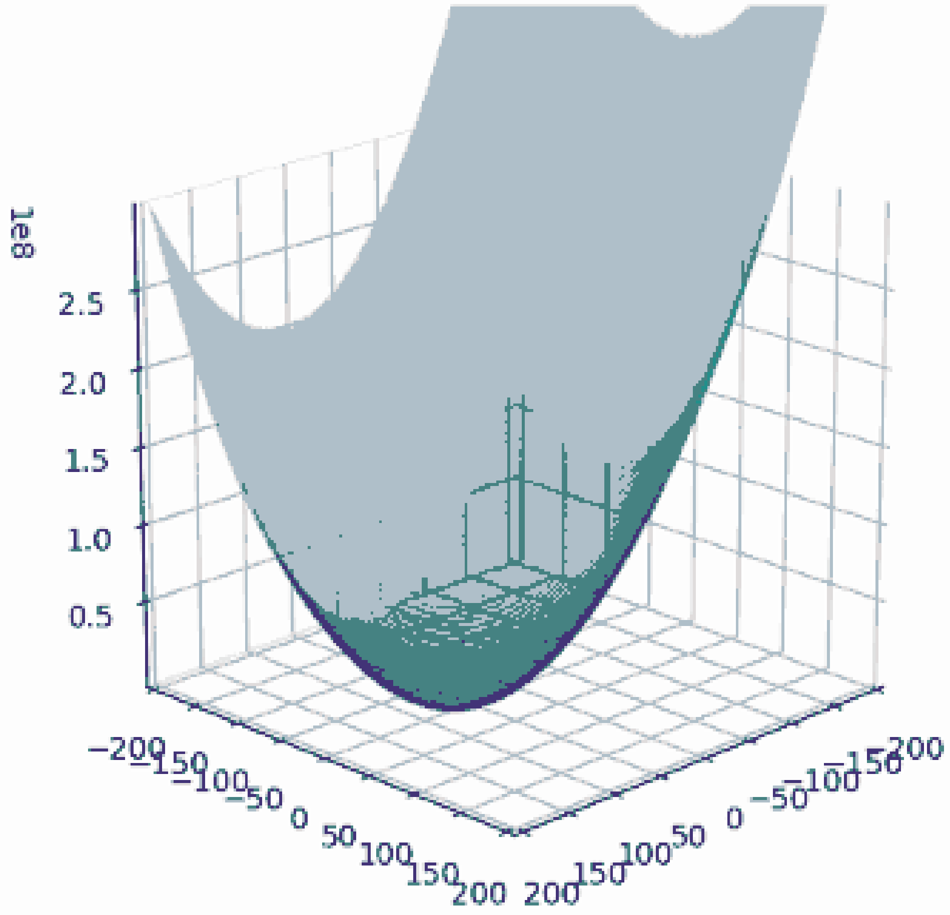
- Contour Plots
- Feature Scaling

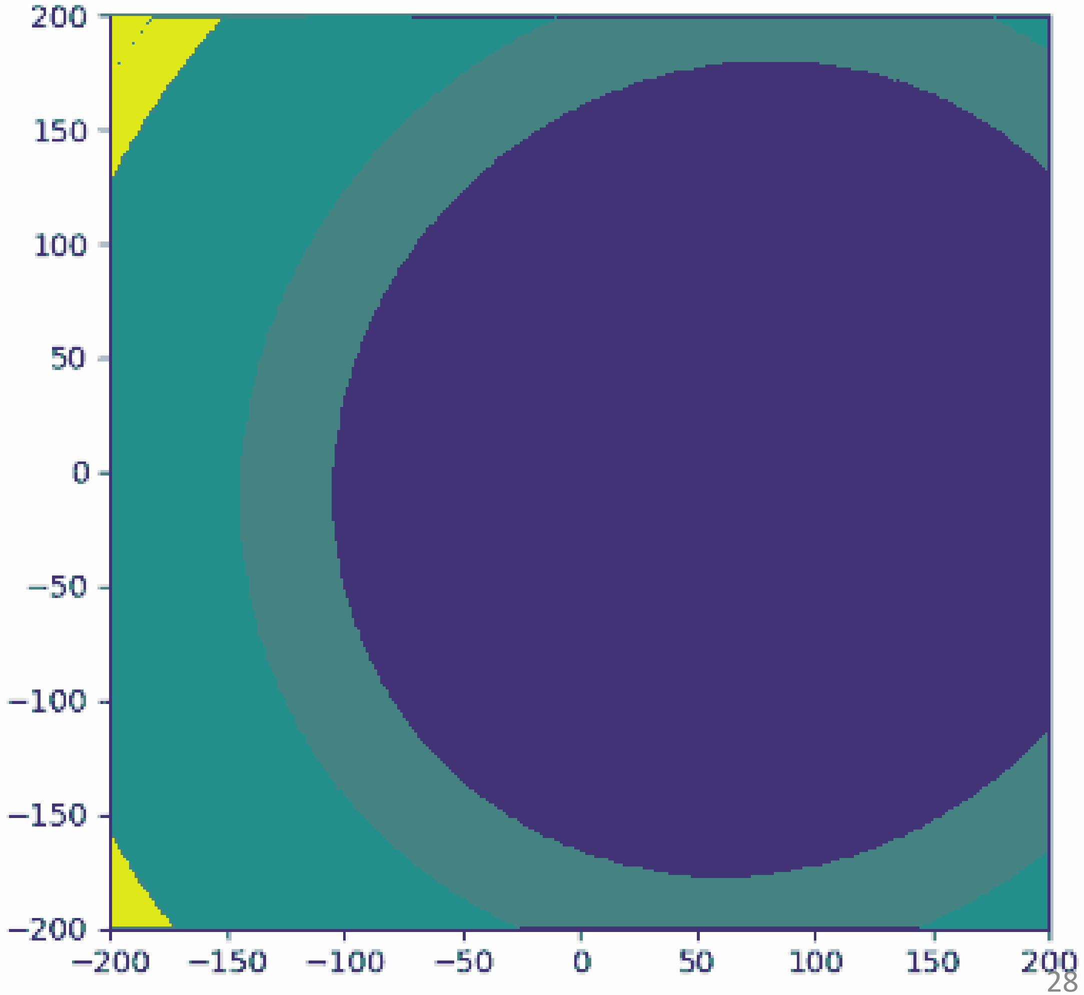
Linear Regression (LR) & Loss Function



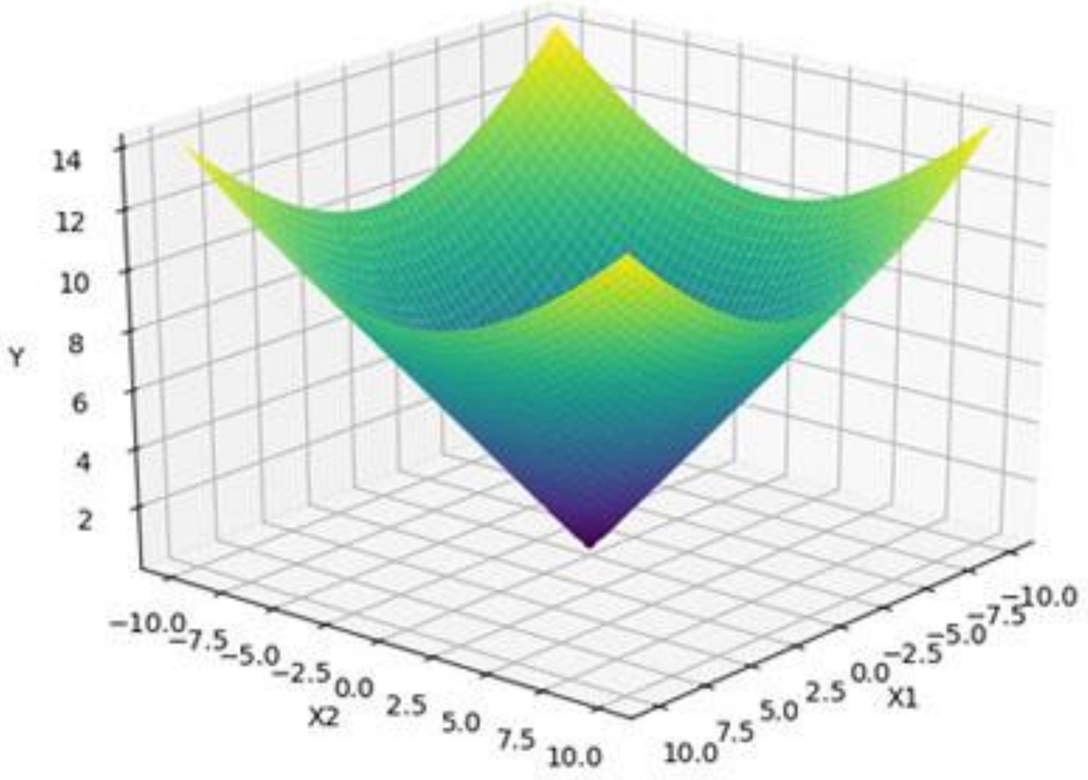


Contour Plot

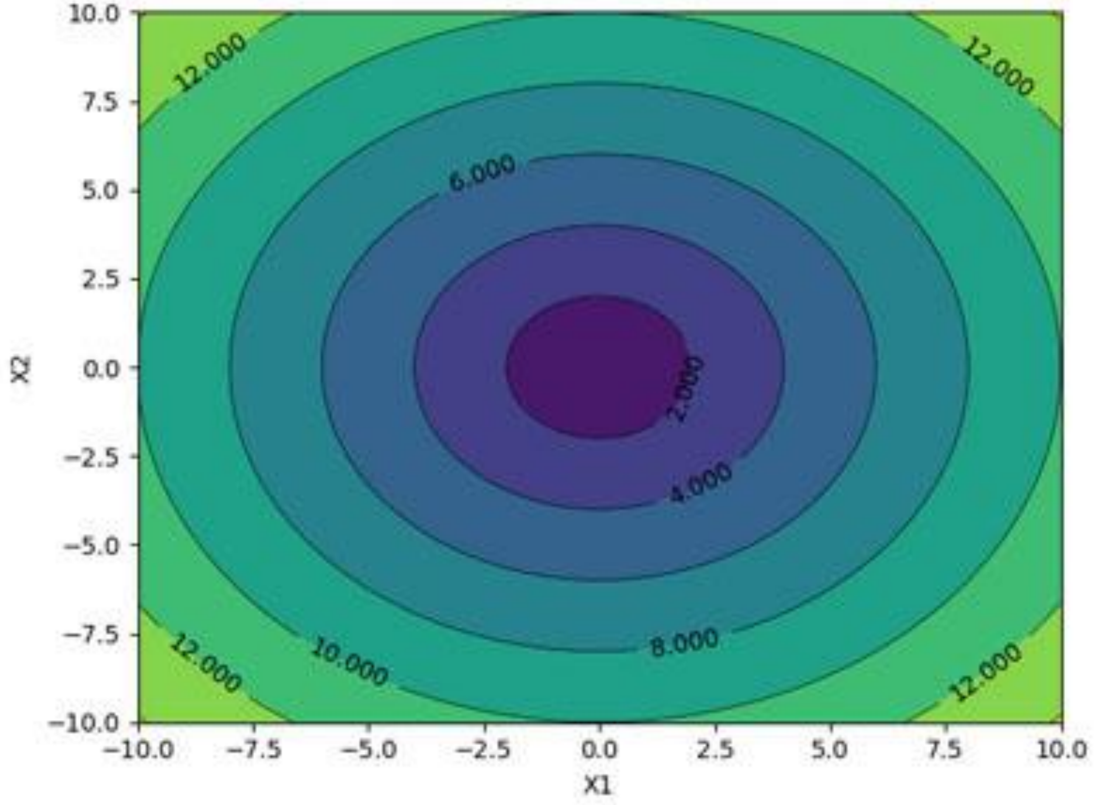




Contour Plot

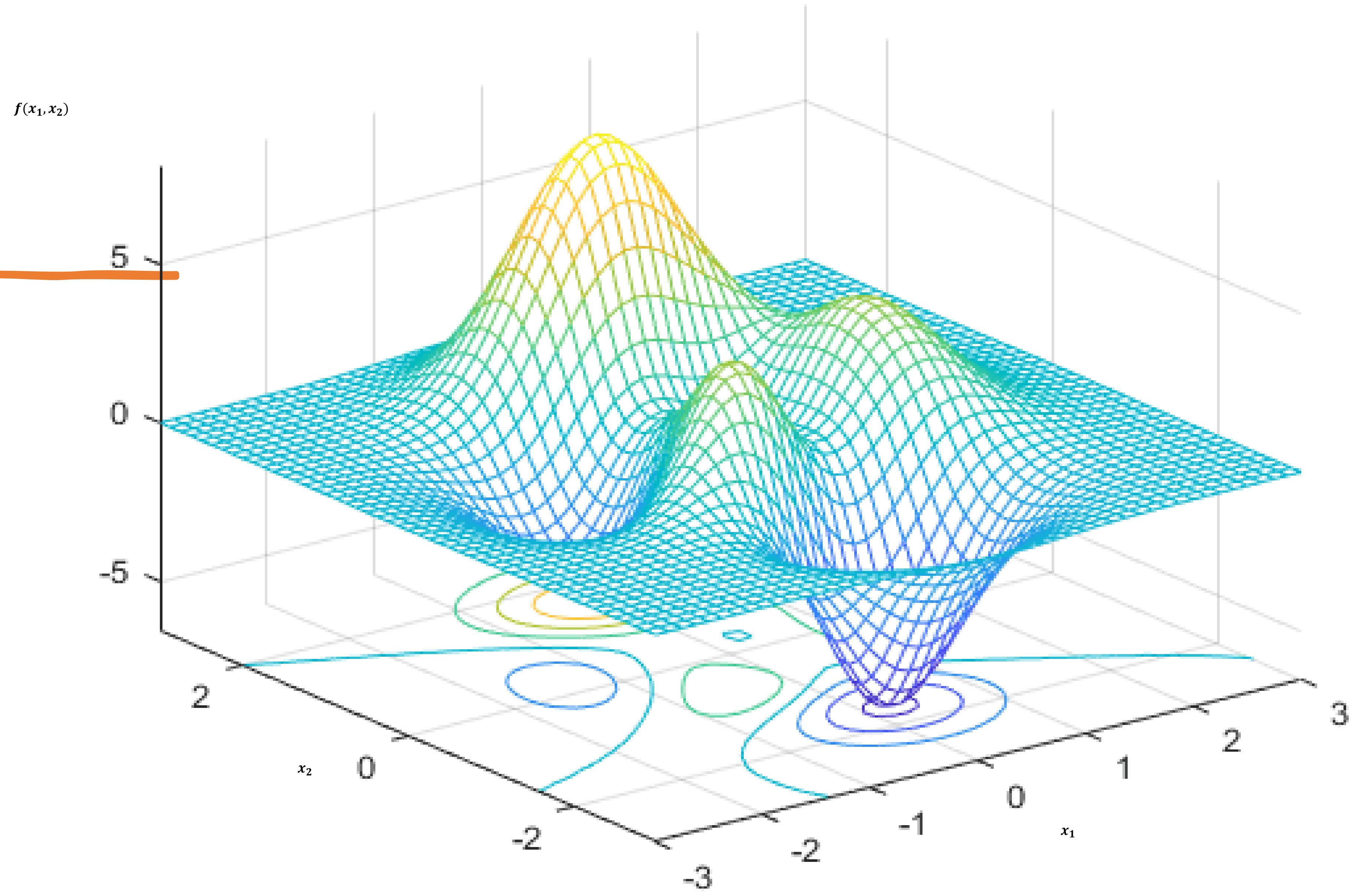


3D Plot

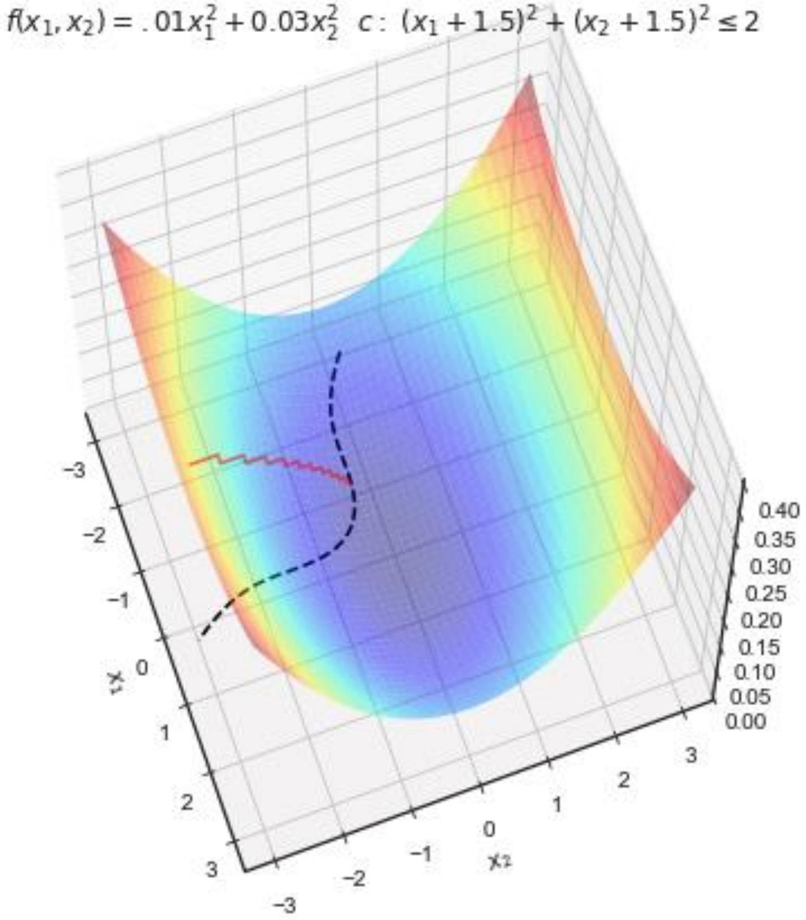


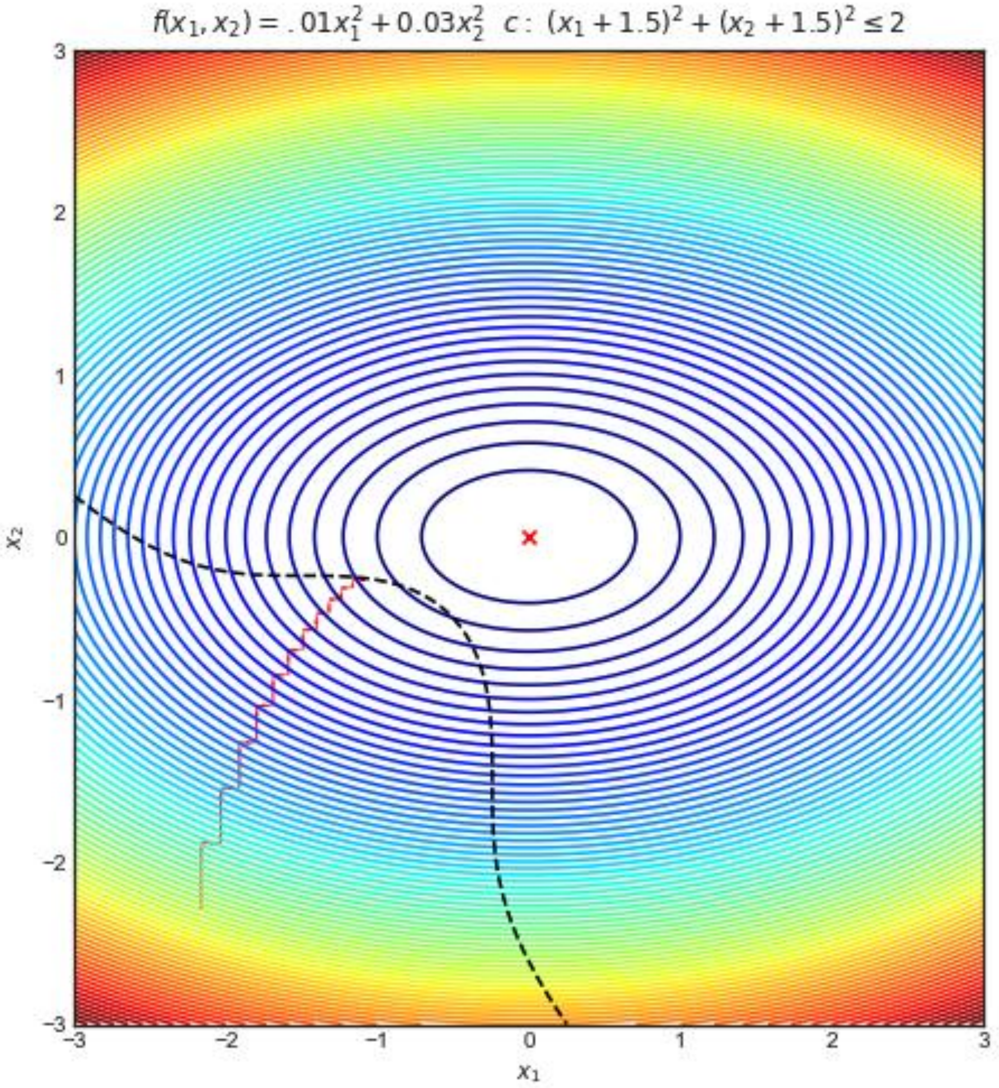
Contour Plot

Contour Plot



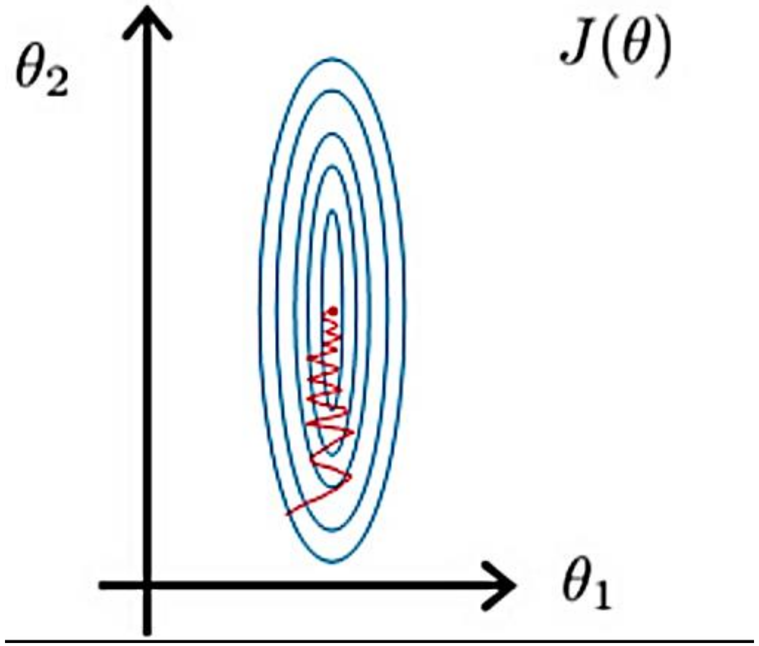
Contour Plot





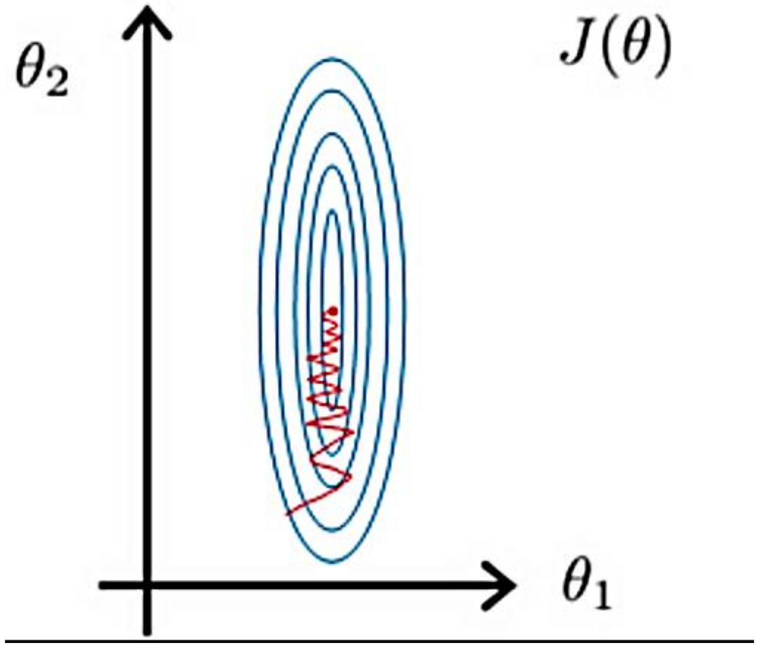
Features Scaling

- **The Problem:**
  - If we have two features, x_1 with a large numerical scale and x_2 with a small scale, then the corresponding coefficient θ_1 typically has a smaller magnitude, while θ_2 has a larger magnitude, in order to produce comparable contributions to the prediction.
  - Since each component of the gradient depends on the feature values, the gradient component associated with the large-scale feature tends to be much larger than that of the small-scale feature.



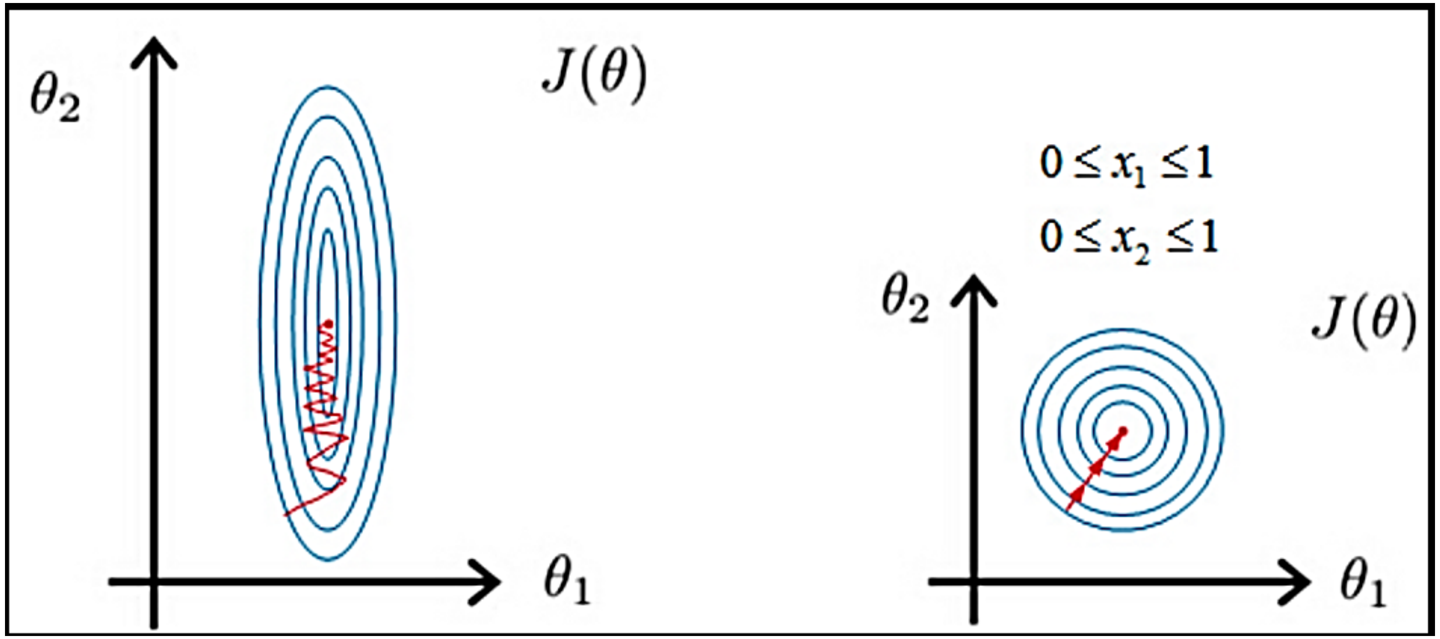
Features Scaling

- **The Problem:**
  - This implies a small range of $\boldsymbol{\theta_1}$ with large update in its direction and large range of $\boldsymbol{\theta_2}$ with small update in its direction.
  - This makes the gradient descent oscillates during training and consumes large number of iterations.
  - This leads slow or unstable convergence.



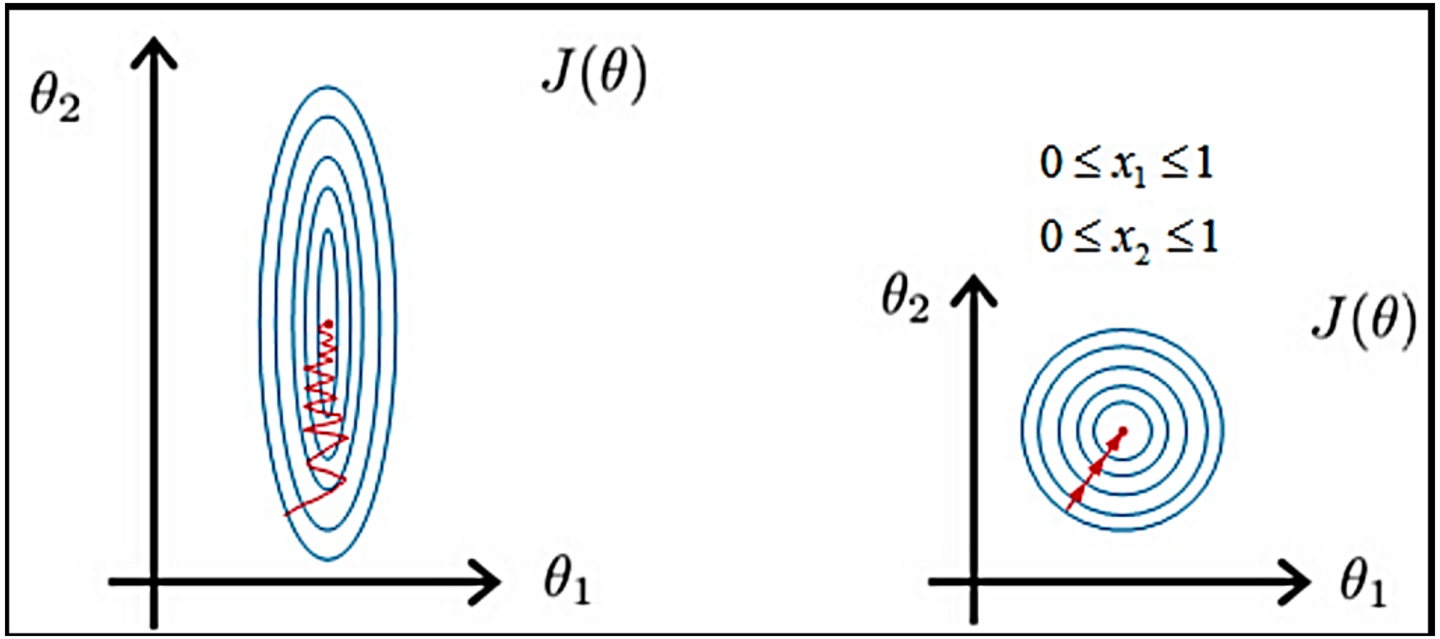
Features Scaling

- **Solution:**
  - It is generally a good practice to scale features to a similar range before training a model.
  - Ensure that features are on similar scale.
  - For gradient-based algorithms, features scaling improves the convergence speed and reliability.



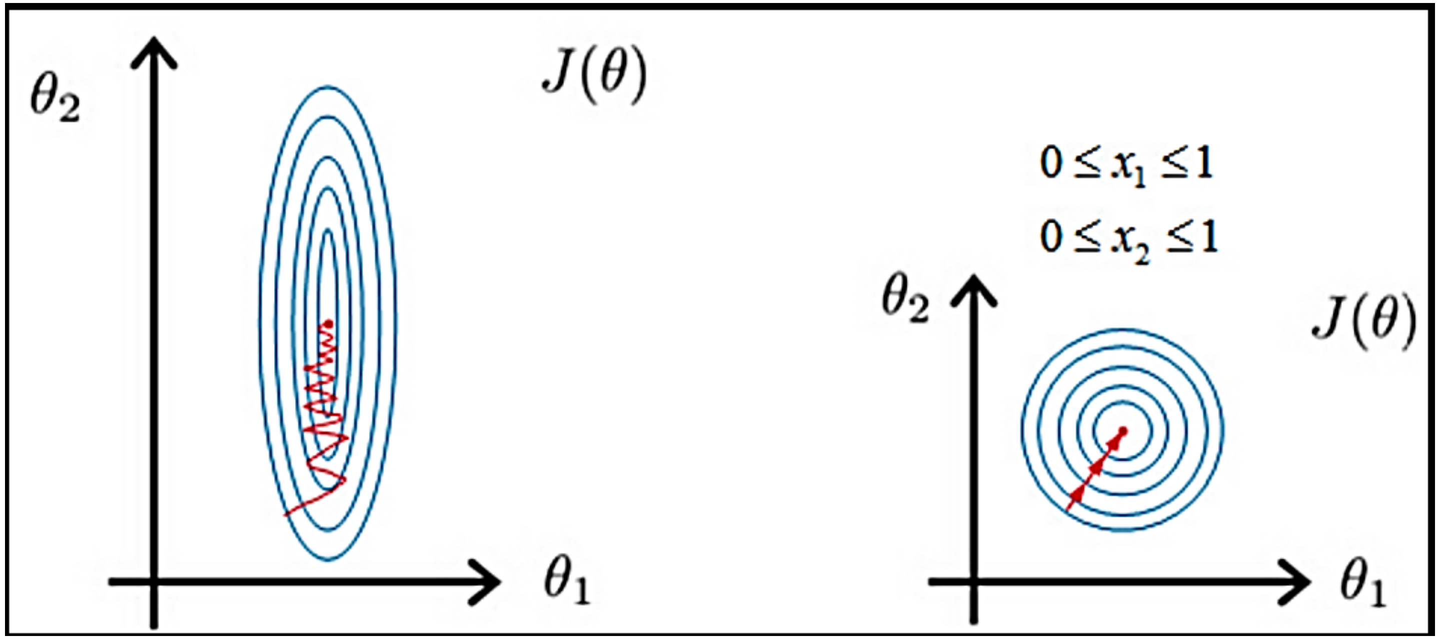
Features Scaling

- **Note:**
  - For **gradient-based** algorithms, features scaling improves the convergence speed.
  - **Distance-based** algorithms like **KNN, K-means,** and **SVM** are most affected by the range of features.
  - **Tree-based** algorithms, on the other hand, are fairly **insensitive** to the scale of the features.



Features Scaling

- **Recommendation:**
  - Use feature scaling when the algorithm calculates distances (K-Nearest Neighbor and Support Vector Machines) or is trained with Gradient Descent (Regression).



Features Scaling

- **Min-Max Normalization:** (Sometimes just called normalization)
  - It scales each variable/feature in the [0,1] range.
  - This method preserves the shape of the original distribution and is sensitive to outliers.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

from sklearn.preprocessing import MinMaxScaler

Features Scaling

- **Mean Normalization:** (Sometimes just called standardization)
  - It produces a distribution centered at 0 with a standard deviation of 1.
  - The data will be scaled to a small interval.
  - Sensitive to outliers

$$x' = \frac{x - \bar{x}}{\sigma}$$

```
from sklearn.preprocessing import StandardScaler
```

Features Scaling:

- **Robust Scaling**
  - All distributions have most of their densities around 0 and a shape that is more or less the same.
  - The interquartile range (IQR) makes this method robust to outliers (hence the name).

```
from sklearn.preprocessing import RobustScaler
```

$$x' = \frac{x - Q_2(x)}{Q_3(x) - Q_1(x)}$$

where Q are quartiles.

Resources

- https://www.coursera.org/learn/machine-learning
- https://machinelearningmastery.com/analytical-vs-numerical-solutions-in-machine-learning/
- https://www.youtube.com/watch?v=e6kf6DDQVYA&ab_channel=TreeSoftMatterTheory
- https://en.wikipedia.org/wiki/Mathematical_optimization
- https://builtin.com/data-science/gradient-descent
- https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a
- https://math.stackexchange.com/questions/2202545/why-using-squared-distances-in-the-cost-function-linear-regression
- https://towardsdatascience.com/optimization-loss-function-under-the-hood-part-ii-d20a239cde11
- https://www.mathsisfun.com/gradient.html
- https://en.wikipedia.org/wiki/Derivative
- https://www.mathsisfun.com/calculus/derivatives-introduction.html
- https://math.libretexts.org/Bookshelves/Calculus/Map%3A_Calculus__Early_Transcendentals_(Stewart)/14%3A_Partial_Derivatives/14.01%3A_Functions_of_Several_Variables
- https://slideplayer.com/slide/4753135/
- https://en.wikipedia.org/wiki/Gradient
- https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/partial-derivative-and-gradient-articles/a/the-gradient
- https://la.mathworks.com/help/matlab/ref/meshc.html
- http://www.adeveloperdiary.com/data-science/how-to-visualize-gradient-descent-using-contour-plot-in-python/
- https://rpubs.com/mgswiss15/M6C_7Multivariate
- https://stats.stackexchange.com/questions/354046/coordinate-descent-with-constraints
- https://www.mathworks.com/help/optim/ug/local-vs-global-optima.html#:~:text=A%20local%20minimum%20of%20a,at%20all%20other%20feasible%20points.
- https://en.wikipedia.org/wiki/Maxima_and_minima
- https://wngaw.github.io/linear-regression/
- http://www.cheerml.com/saddle-points
- https://towardsdatascience.com/understand-convexity-in-optimization-db87653bf920
- https://towardsdatascience.com/understand-convexity-in-optimization-db87653bf920
- https://www.sciencedirect.com/topics/engineering/convex-function
- https://www.math24.net/convex-functions#example2
- https://tutorial.math.lamar.edu/Classes/CalcI/NewtonsMethod.aspx
- https://en.wikipedia.org/wiki/Newton's_method
- https://tutorial.math.lamar.edu/Classes/CalcI/NewtonsMethod.aspx

Resources

- https://realpython.com/linear-regression-in-python/
- https://towardsdatascience.com/linear-regression-using-python-b136c91bf0a2
- https://towardsdatascience.com/why-norms-matters-machine-learning-3f08120af429
- https://towardsdatascience.com/why-norms-matters-machine-learning-3f08120af429
- https://machinelearningmastery.com/vector-norms-machine-learning/
- https://medium.com/linear-algebra/part-18-norms-30a8b3739bb
- https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0
- Andrew Ng, Machine Learning, Stanford University, Coursera
- https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0
- https://medium.com/data-science-365/linear-regression-with-gradient-descent-895bb7d18d52
- https://www.holehouse.org/mlclass/17_Large_Scale_Machine_Learning.html
- https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220
- https://towardsdatascience.com/machine-learning-fundamentals-via-linear-regression-41a5d11f5220
- https://www.analyticsvidhya.com/blog/2019/08/detailed-guide-7-loss-functions-machine-learning-python-code/
- https://builtin.com/data-science/gradient-descent
- https://www.mltut.com/stochastic-gradient-descent-a-super-easy-complete-guide/
- https://towardsdatascience.com/linear-regression-using-gradient-descent-97a6c8700931
- https://kaigangi72.medium.com/stochastic-gradient-descent-demystified-part-1-8e4b897079b7
- https://medium.datadriveninvestor.com/gradient-descent-algorithm-b4c5afb4eb98
- https://medium.com/mindorks/an-introduction-to-gradient-descent-7b0c6d9e49f6
- https://medium.com/@venkatavinay222/at-the-end-machine-learning-is-all-about-optimization-ft-gradient-descent-e1588b7d95d2
- “Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow” by Aurélien Géron
- https://laptrinhx.com/feature-scaling-why-and-how-3308094292/
- https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3