COMP2261 ARTIFICIAL INTELLIGENCE / MACHINE LEARNING

Gradient Descent
-- Learning Algorithm

Dr SHI Lei

Durham University
learning lab

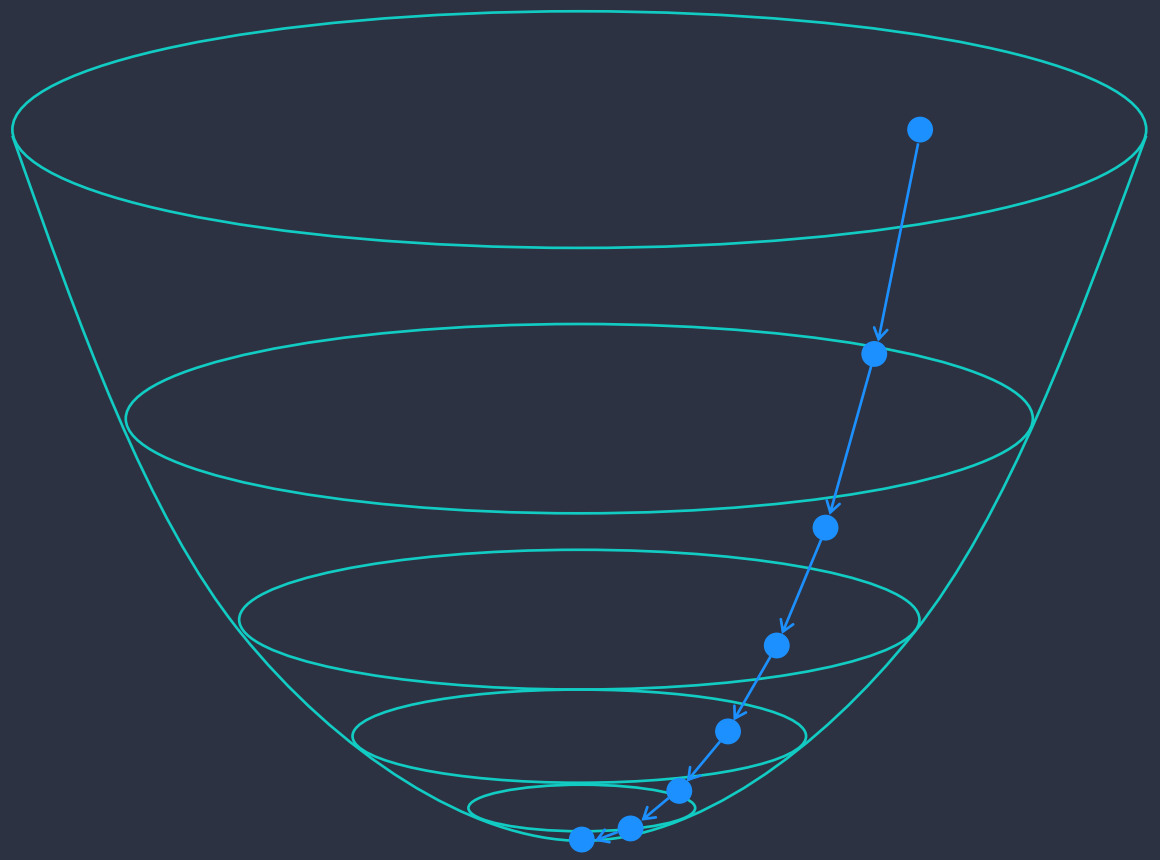
Cost Function

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

- Cost Function $J(\boldsymbol{\theta})$ is a function of model parameter vector $\boldsymbol{\theta} = \begin{bmatrix}\theta_0\\ \theta_1\\ \theta_2\\ \cdots\\ \theta_n\end{bmatrix}$ (independent variables).
- Hypothesis function $h_{\boldsymbol{\theta}}(x) = \boldsymbol{\theta}^T\boldsymbol{x} = [\boldsymbol{\theta_0}, \theta_1, \boldsymbol{\theta_2}, \ldots, \theta_n]\begin{bmatrix}1\\ x_1\\ x_2\\ \cdots\\ x_n\end{bmatrix} = \theta_0 + \theta_1 \cdot x_1 + \theta_2 \cdot x_2 + \cdots + \theta_n \cdot x_n$

  is the model to make predictions.
- Goal: find $\boldsymbol{\theta}$ to $\underset{\boldsymbol{\theta}}{minimise}\, J(\boldsymbol{\theta})$.

Gradient Descent



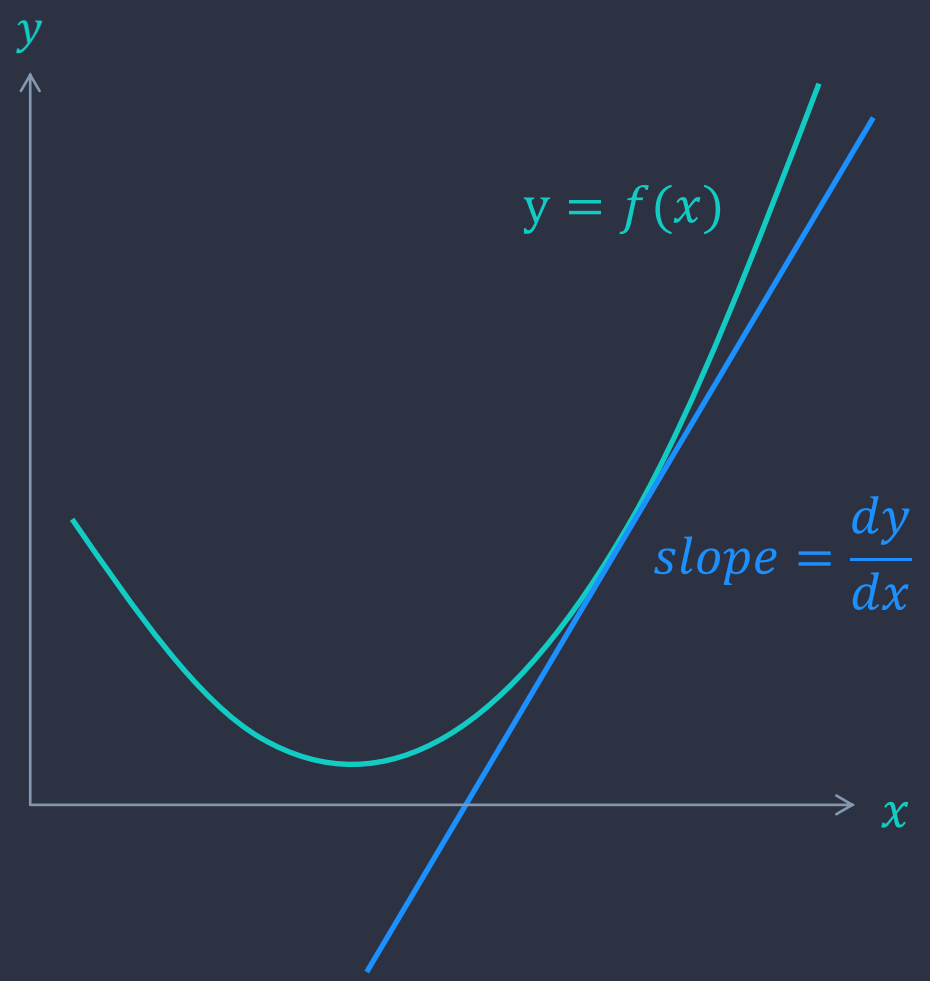
- In order to find the model parameter vector that minimises the cost function $J(\theta)$, we iteratively try different θs in our learning algorithm, and gradient descent can help decide which θs to try and when to stop.

- In order for a gradient descent learning algorithm to converge, i.e. reach the minimum of the cost function, as quickly as possible, we need to change the model parameters in the direction of the steepest slope.

Learning Objectives

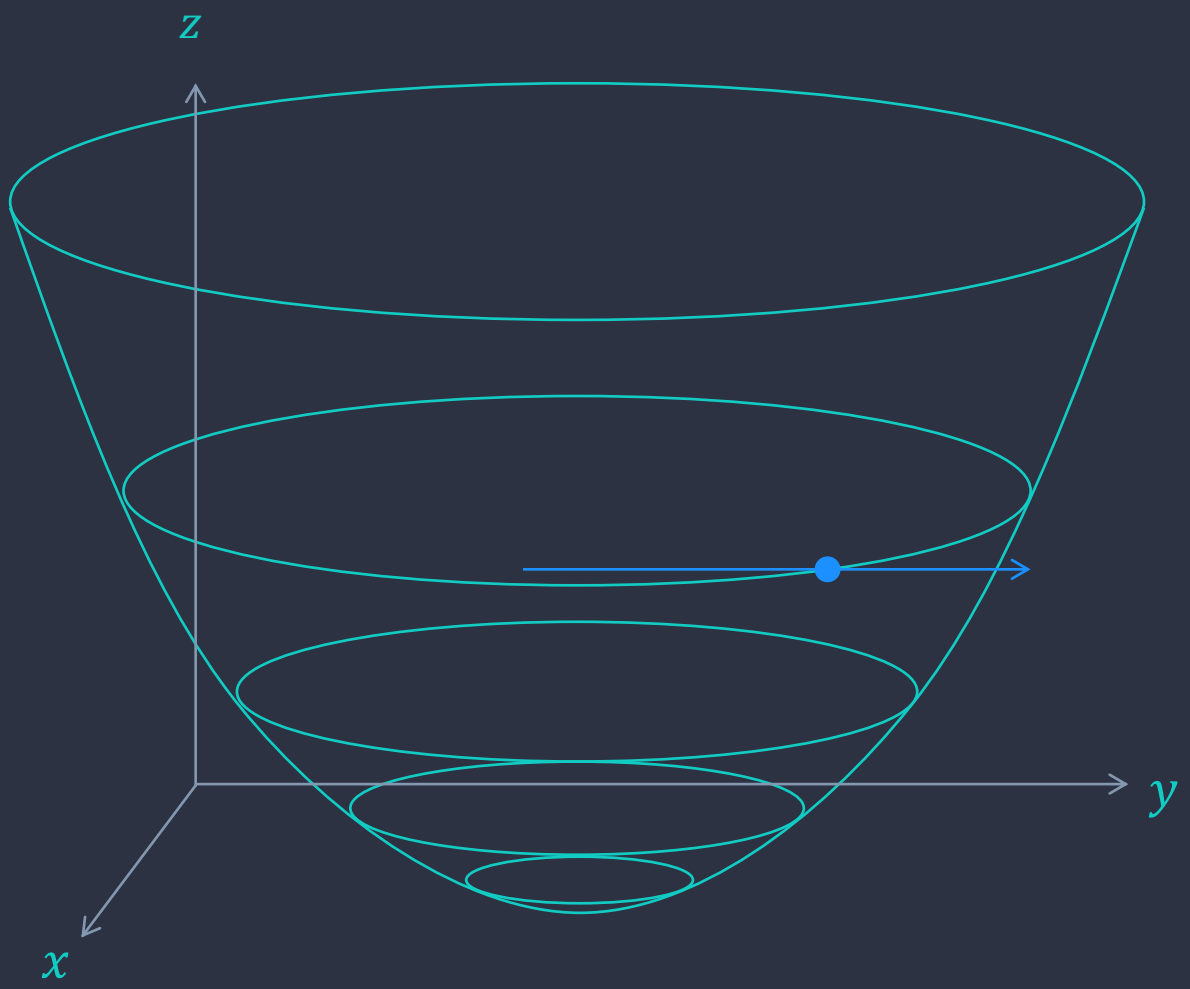
- Understand the concept of gradient descent.
- Understand how the gradient descent algorithm works.

Partial Derivative and Gradient



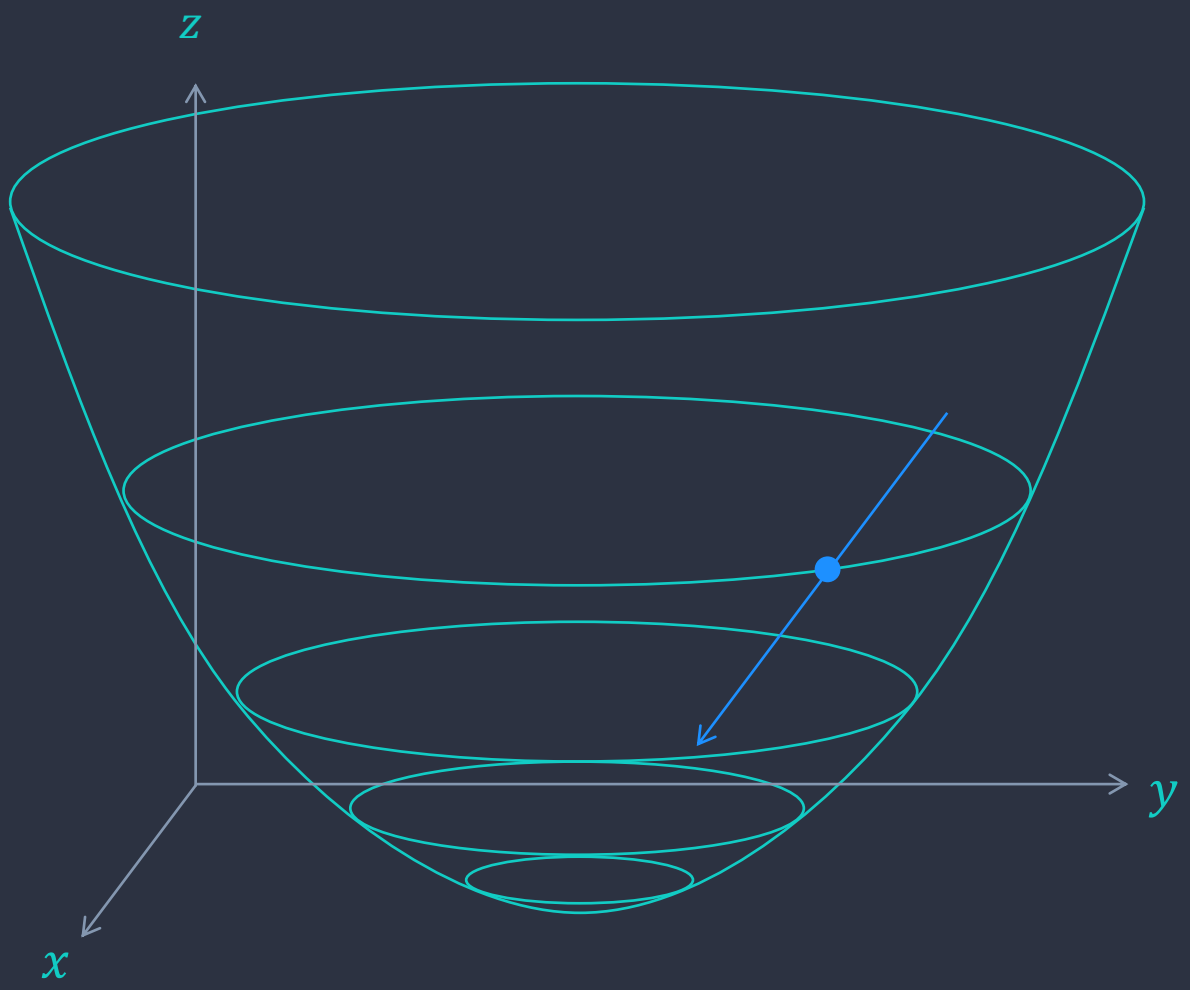
- Derivative at a point is the slope of the curve at that point, and it measures the steepness of the curve at that point.
- Derivative of y with respect to x : $\frac{dy}{dx}$

Partial Derivative and Gradient



- For a surface, we measure the steepness at a point in specific directions independently, e.g. in the direction of x and in the direction of y.
- Partial Derivative of z with respect to x : $\frac{\partial z}{\partial x}$

Partial Derivative and Gradient



- Partial Derivative of z with respect to x : $\frac{\partial z}{\partial x}$
- Partial Derivative of z with respect to y : $\frac{\partial z}{\partial y}$
- Gradient of z: $\text{grad z} = \frac{\partial}{\partial x}\vec{i} + \frac{\partial}{\partial y}\vec{j}$
- Gradient can be considered as the slop in higher dimensions. It points in the direction of maximum change of the function.

$$\nabla z = < \frac{\partial z}{\partial x}, \frac{\partial z}{\partial y} >$$

Gradient Descent

Univariate Linear Regression model $h_{\boldsymbol{\theta}}(x) = \boldsymbol{\theta}^T \boldsymbol{x} = [\boldsymbol{\theta_0}, \boldsymbol{\theta_1}] \begin{bmatrix} 1 \\ x \end{bmatrix} = \boldsymbol{\theta_0} + \boldsymbol{\theta_1} \cdot \boldsymbol{x}$

(one independent variable and one dependent variable)

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

predicted label actual label

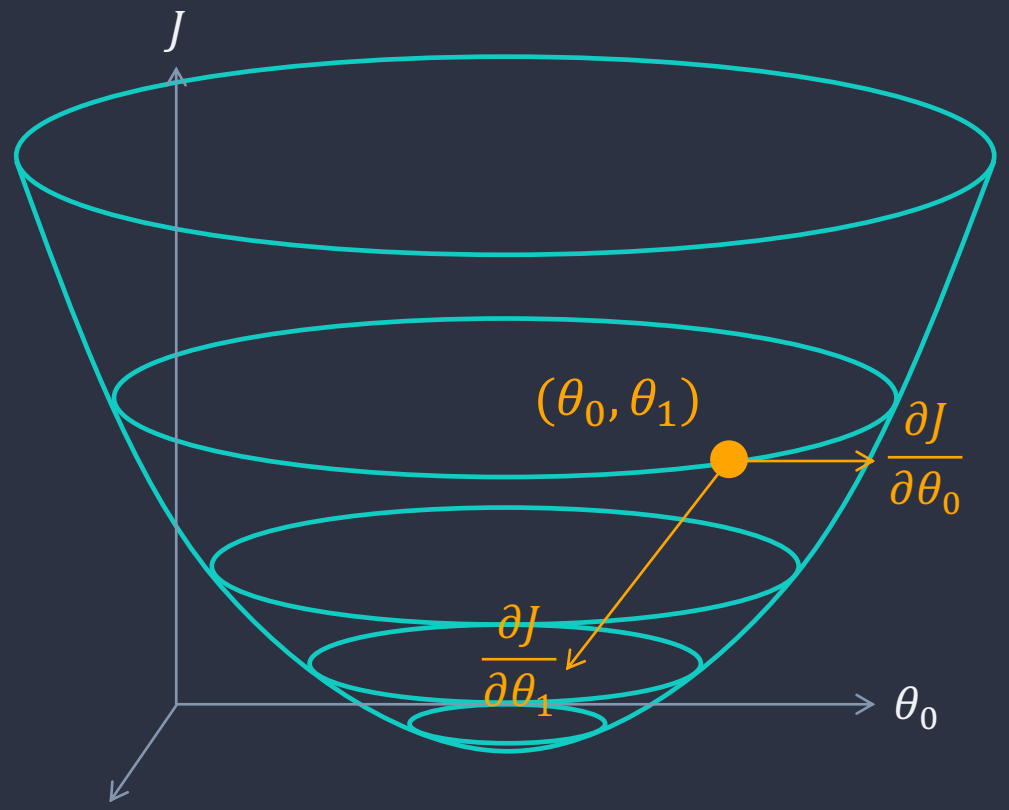
$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(\boxed{\theta_0 + \theta_1 \cdot x^{(i)}} - y^{(i)})^2 = \frac{1}{2m}\sum_{i=1}^{m}(\boxed{h_\theta(x^{(i)})} - y^{(i)})^2$$

$$\frac{\partial}{\partial\theta_0}J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}(\boxed{\theta_0 + \theta_1 \cdot x^{(i)}} - y^{(i)}) \implies \frac{\partial}{\partial\theta_0}J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}(\boxed{h_\theta(x^{(i)})} - y^{(i)})$$

$$\frac{\partial}{\partial\theta_1}J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}(\boxed{\theta_0 + \theta_1 \cdot x^{(i)}} - y^{(i)}) \cdot x^{(i)} \implies \frac{\partial}{\partial\theta_1}J(\theta_0, \theta_1) = \frac{1}{m}\sum_{i=1}^{m}(\boxed{h_\theta(x^{(i)})} - y^{(i)}) \cdot x^{(i)}$$

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

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

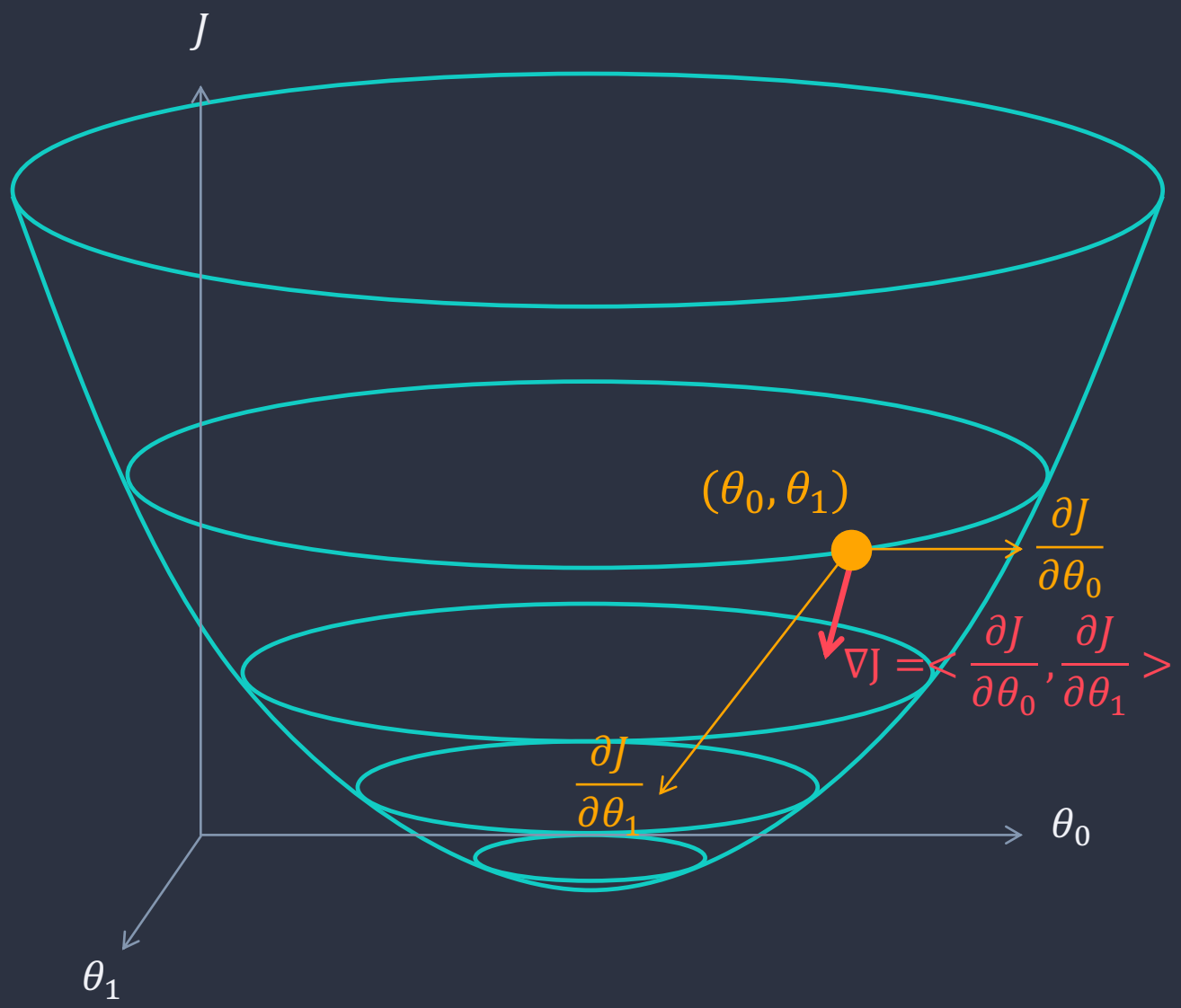


predicted label actual label actual feature

all constants (the i-th instance in training set)

Gradient: $\nabla J = < \frac{\partial J}{\partial\theta_0}(\theta_0,\theta_1), \frac{\partial J}{\partial\theta_1}(\theta_0,\theta_1) >$

Note: $\frac{\partial J}{\partial\theta_0}$ *is the same as* $\frac{\partial J}{\partial\theta_0}(\theta_0,\theta_1)$

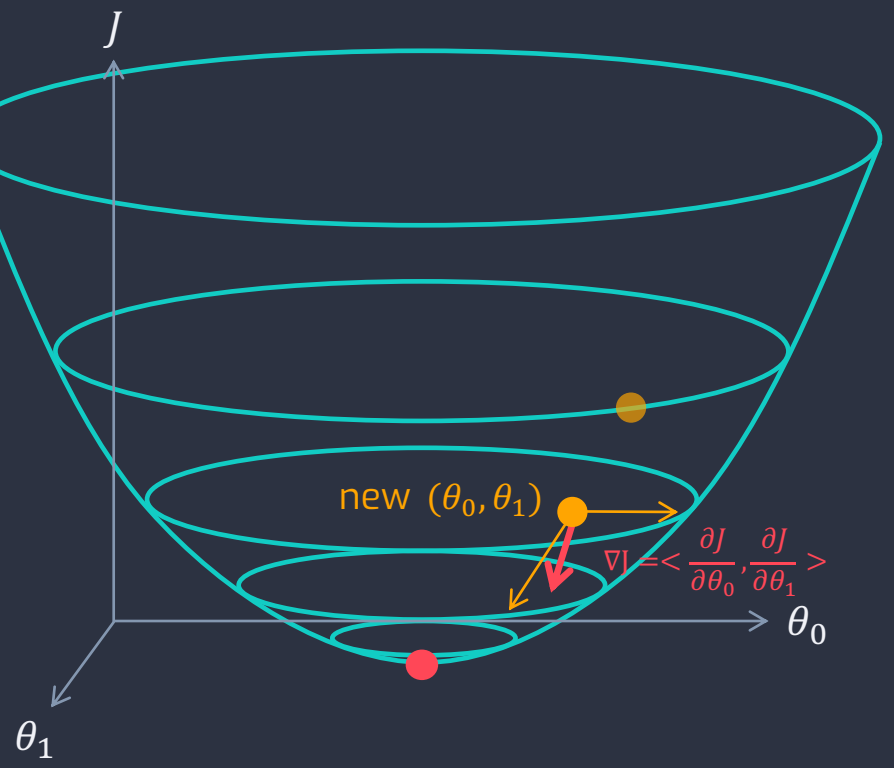


$$\theta_0 := \theta_0 - \alpha \cdot \frac{\partial J}{\partial \theta_0}(\theta_0, \theta_1)$$

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

Learning Rate

(hyperparameter of the learning algorithm)



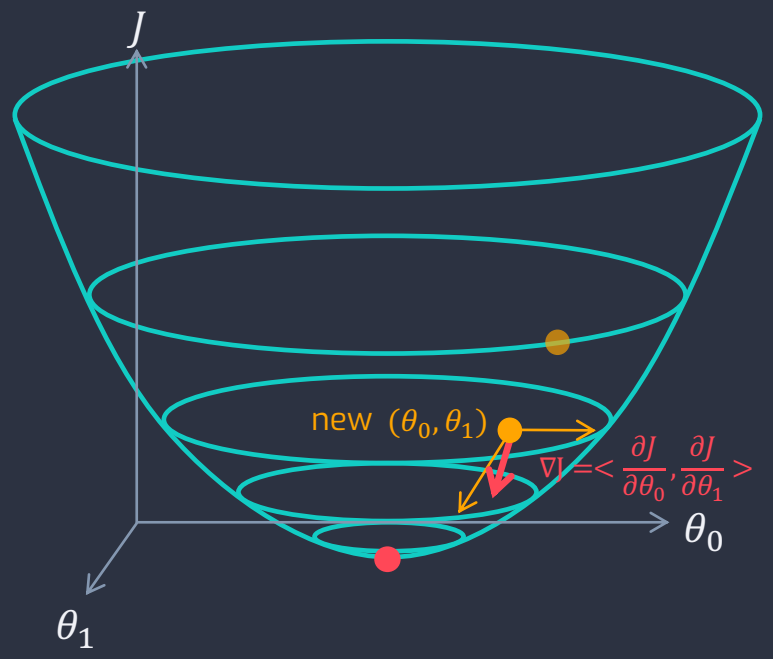
Repeat until convergence {

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

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

}

- Learning rate α controls how big a step our learning gradient descent learning algorithm is moving towards a minimum.
  - Large α: learning algorithm is very aggressive (huge steps to a minimum).
  - Small α: learning algorithm only takes tiny steps towards a minimum.



Repeat until convergence {

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

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

}

- To update all these model parameters (θs) simultaneously.

correct: simultaneously update

$$temp_0 := \theta_0 - \alpha \cdot \frac{\partial J}{\partial \theta_0}(\theta_0, \theta_1)$$

$$temp_1 := \theta_1 - \alpha \cdot \frac{\partial J}{\partial \theta_1}(\theta_0, \theta_1)$$

$$\theta_0 := temp_0$$

$$\theta_1 := temp_1$$

incorrect: not simultaneously update

$$temp_0 := \theta_0 - \alpha \cdot \frac{\partial J}{\partial \theta_0}(\theta_0, \theta_1)$$

$$\theta_0 := temp_0$$

$$temp_1 := \theta_1 - \alpha \cdot \frac{\partial J}{\partial \theta_1}(\theta_0, \theta_1)$$

$$\theta_1 := temp_1$$

Takeaway Points

- Partial derivatives of the cost function with respect to each model parameters to calculate gradient
- Simultaneously to update all model parameters.
- Updating iterations stop when the algorithm converges.