📕机器学习--微积分

[课程链接: Coursera]
[课程链接: Bilibili]
[课程链接: Github] -- Homework

Week 1: Derivatives and Optimization

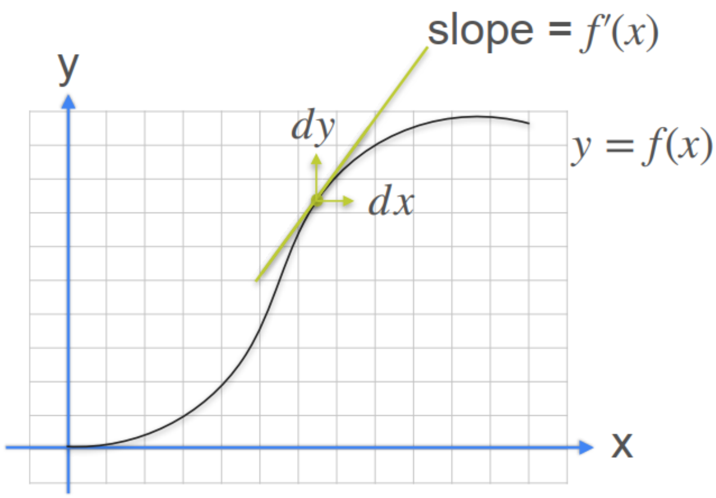
1.1 导数的表示方法： Derivative notation

$y = f(x)$

Derivative of f is expressed as:

$f'(x)$ **Lagrange's notation**

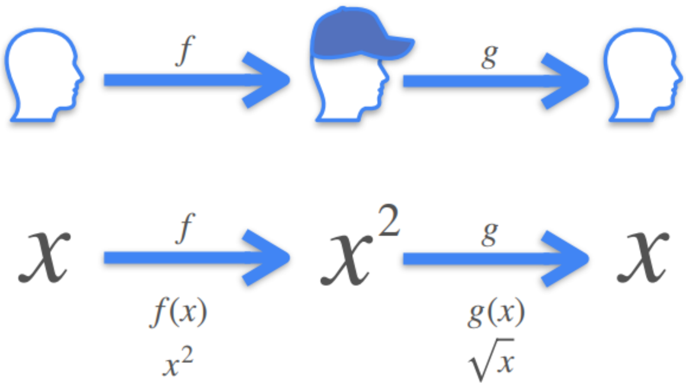
$\frac{dy}{dx} = \frac{d}{dx}f(x)$ **Leibniz's notation**



> 上述分别是拉格朗日表示法和莱布尼茨表示法

1.2 反函数：Inverse Function

What's an inverse?



$g(x)$ and $f(x)$ are inverses

$g(x) = f^{-1}(x)$

$g(f(x)) = x$

$\sqrt{x^2} = x$ for $x > 0$

> 反函数的性质：
>
> 如果f(x)和g(x)互为反函数：
>
> - **性质1：g(f(x)) = x**
> - **性质2：如果(a, b)在f(x)上，则(b, a)在g(x)上**
> - **性质3：如果(a, b)在f(x)上，且f(x)在x = a处的导数为m，则g(x)在x = b处的导数为1/m**
>
> 例如：
>
> $$x^2和sqrt(x)互为反函数$$
>
> $$e^x和ln(x)互为反函数$$

1.3 欧拉数：e

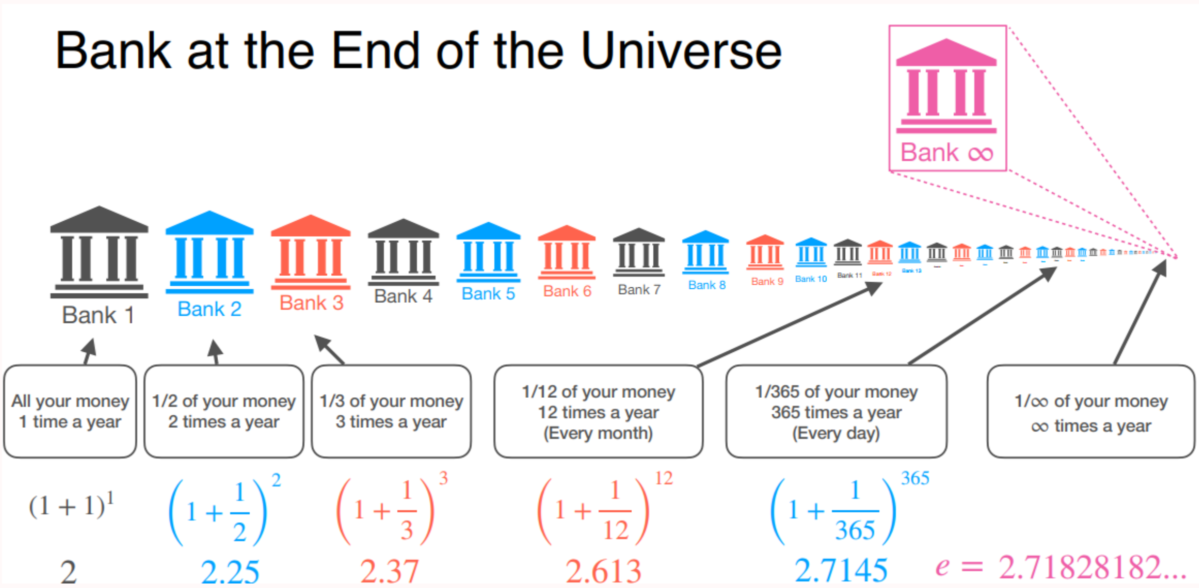
$e = 2.71828182\ldots$

n	1	10	100	1000	∞
$\left(1+\frac{1}{n}\right)^n$	2	2.594	2.705	2.717	e

$$f(x) = e^x$$

$$f'(x) = e^x$$

🙋银行存款中的应用：



1.4 导数的存在性：Existence of the derivative

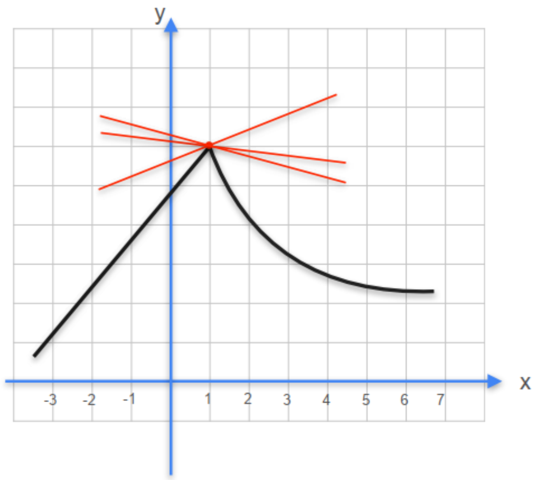
For a function to be differentiable (可微) at a point:

- The derivative has to exist for that point

For a function to be differentiable (可微) at an interval:

- The derivative has to exist for *every* point in the interval

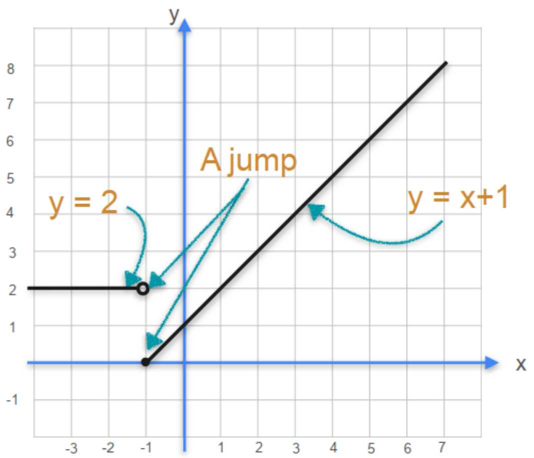
1. **Generally, when a fuction has a corner (角) or a cusp (尖), the function is not differentiable at that point.**



At which point in this function does the derivative not exist?

The entire function is non-differentiable because a derivative does not exist for all points in the domain.

2. 不连续的函数是不可微的



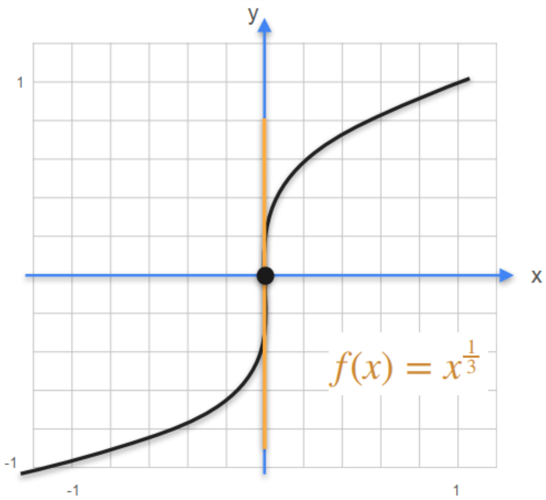
This is a piece-wise function.

$$f(x) = \begin{cases} 2, & \text{if } x < -1 \\ x+1, & \text{if } x \geq -1 \end{cases}$$

Jump Discontinuity

The graph of the function does not appear to be continuous

3. 切线和y轴平行的函数也是不可微的

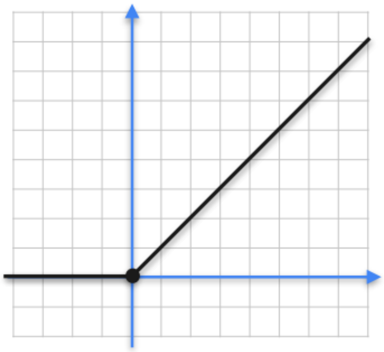


Vertical tangents

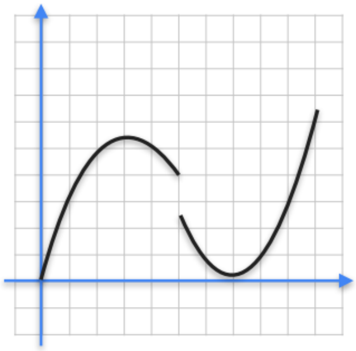
At x = 0, this graph has a tangent line that runs straight up parallel to the y-axis

关于切线的一个知乎回答：切线可以看做一小段曲线的近似

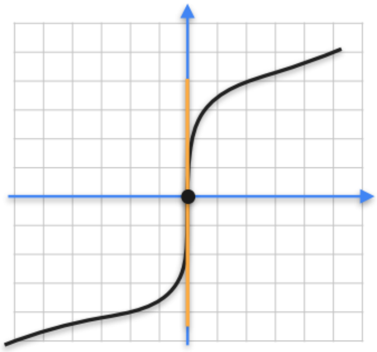
⭐总结：有尖点，不连续，有垂直切线的函数不可微



Corners/Cusps

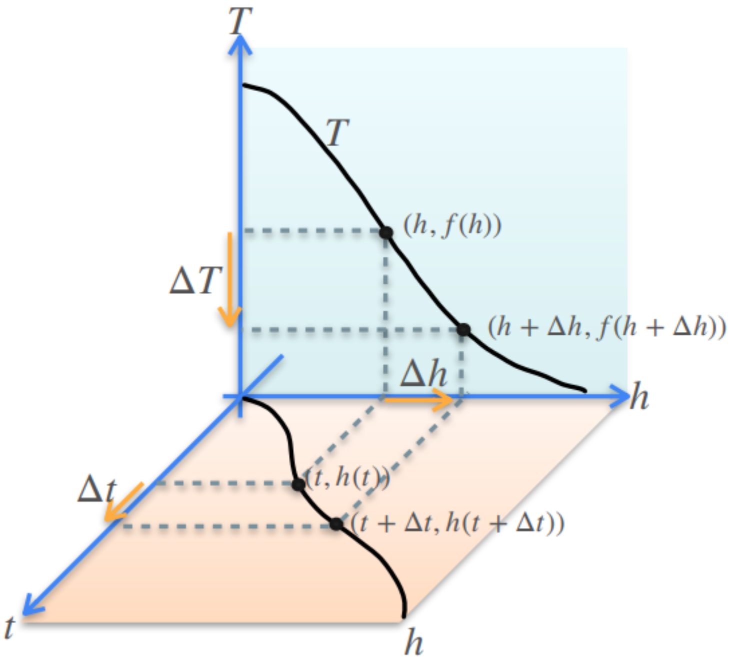


Jump Discontinuity



Vertical tangents

1.5 Chain Rule

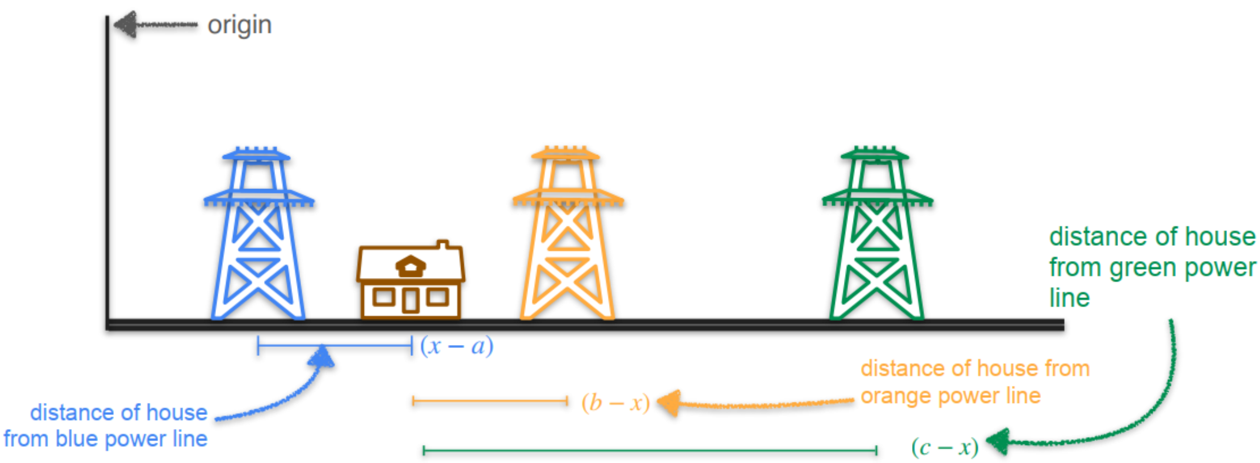


$$\Delta t \rightarrow \Delta h \rightarrow \Delta T$$

$$\frac{\Delta T}{\Delta t} = \frac{\Delta T}{\Delta h}\frac{\Delta h}{\Delta t}$$

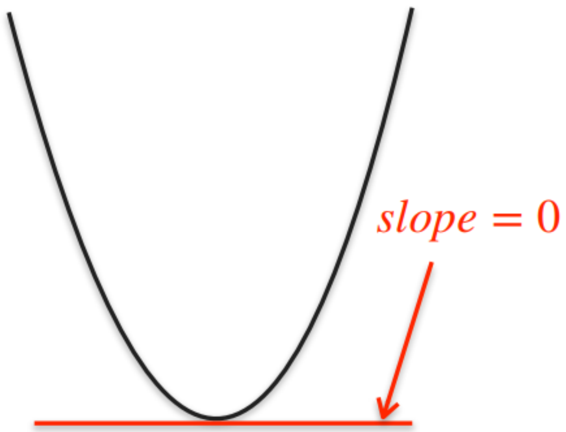
1.6 平方损失优化

- Question：已知有三个电塔，距离原点的距离分别是a, b, c，现有一个房子需要确定选址，要求铺设电缆花费最少，其中：铺设电缆的费用与房子与电塔距离的平方成正比



- Solution：

$$(x-a)^2+(x-b)^2+(x-c)^2$$



$$\frac{d}{dx}\left[(x-a)^2+(x-b)^2+(x-c)^2\right]=0$$

$$2(x-a)+2(x-b)+2(x-c)=0$$

$$(x-a)+(x-b)+(x-c)=0$$

$$3x-a-b-c=0$$

$$3x=a+b+c$$

$$x=\frac{a+b+c}{3}$$

平方损失：

$$\text{Minimize } (x-a_1)^2+(x-a_2)^2+\cdots+(x-a_n)^2$$

$$\text{Solution: } x=\frac{a_1+a_2+\cdots+a_n}{n}$$

1.7 对数损失优化

- Question：投10次硬币，7次正面，3次反面则游戏胜利。要求设计一枚特殊的硬币，其正面向上的概率为p，使得游戏获胜的概率最大

p $(1-p)$

Chances of winning: $p^7(1-p)^3=g(p)$

Goal: maximize $g(p)$

- Solution 1：

$$\frac{dg}{dp} = \frac{d}{dp}(p^7(1-p)^3) \overset{\text{Product rule}}{=} \frac{d(p^7)}{dp}(1-p)^3 + p^7 \boxed{\frac{d\left((1-p)^3\right)}{dp}}$$

$$= 7p^6(1-p)^3 + p^7 \boxed{3(1-p)^2(-1)}$$

Chain rule

$$= p^6(1-p)^2\left[7(1-p) - 3p\right]$$

$$= \boxed{p^6}\boxed{(1-p)^2}\boxed{(7-10p)} = 0$$

$$\cancel{p = 0} \qquad \cancel{p = 1} \qquad p = 0.7$$

- Solution 2：另一种更简便的方式是对g(p)取对数，最大化g(p)相当于最大化log(g(p))

$$\log(g(p)) = \log(p^7(1-p)^3) = \log\left(p^7\right) + \log\left((1-p)^3\right)$$

$$= \boxed{7\log(p) + 3\log(1-p) = G(p)}$$

$-G(p)$ is the logloss

$$\frac{dG(p)}{dp} = \frac{d}{dp}(7\log(p) + 3\log(1-p)) = 7\frac{1}{p} + 3\frac{1}{1-p}(-1)$$

$$= \frac{7(1-p) - 3p}{p(1-p)} = 0$$

$$7(1-p) - 3p = 0 \qquad p = 0.7$$

总结：为什么要取对数？

1. Derivative of products is hard, derivative of sums is easy. (一堆式子相乘的导数很难求，但一堆式子相加的导数相对容易)

$$f(p) = p^6(1-p)^2(3-p)^9(p-4)^{13}(10-p)^{500}$$

$$\frac{df}{dp}$$

$$[6p^5](1-p)^2(3-p)^9(p-4)^{13}(10-p)^{500} +$$
$$p^6\left[2(1-p)\right](3-p)^9(p-4)^{13}(10-p)^{500}(-1) +$$
$$p^6(1-p)^2\left[9(3-p)^8\right](p-4)^{13}(10-p)^{500}(-1) +$$
$$p^6(1-p)^2(3-p)^9\left[13(p-4)^{12}\right](10-p)^{500} +$$
$$p^6(1-p)^2(3-p)^9(p-4)^{13}\left[500(10-p)^{499}\right](-1)$$

$$\frac{d}{dp}\log(f)$$

$$\frac{6}{p} + \frac{2}{1-p}(-1) + \frac{9}{3-p}(-1) + \frac{13}{p-4} + \frac{500}{10-p}(-1)$$

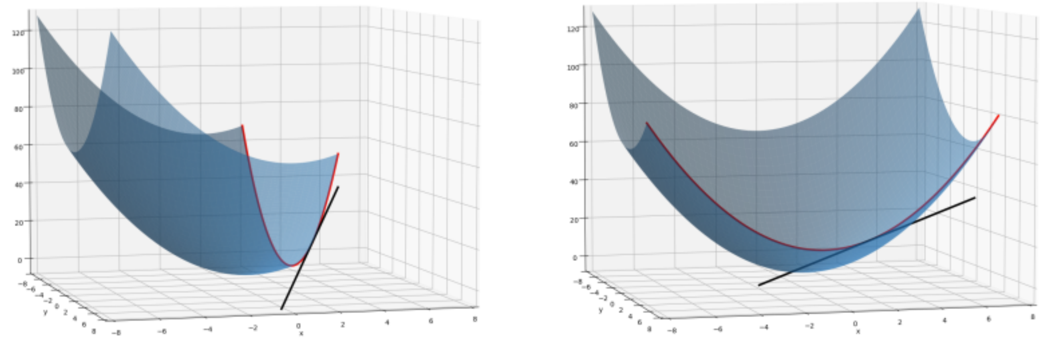
2. Product of lots of tiny things is tiny! (一堆很小的数字相乘会导致很小的结果，计算机可能无法处理，将一个很小的数取对数就可能将这个数变成很大的一个负数)

Week 2: Gradients and Gradient Descent

2.1 Partial derivatives

$$f(x, y) = x^2 + y^2$$

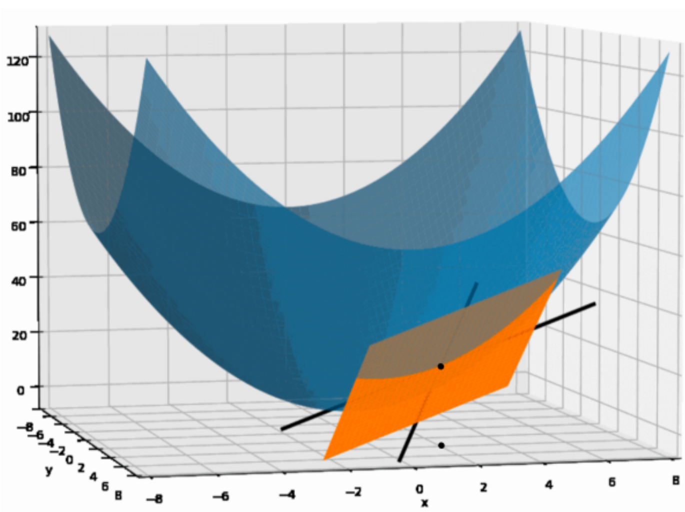
Treat y as a constant　　　　Treat x as a constant



$$\frac{\partial f}{\partial x} = 2x \qquad\qquad \frac{\partial f}{\partial y} = 2y$$

- 求对x的偏导数，就是将y视为常数（用垂直于y轴的平面切割曲面），求f对x的一元导数，对应右图抛物线的导数
- 求对y的偏导数，就是将x视为常数（用垂直于x轴的平面切割曲面），求f对y的一元导数，对应左图抛物线的导数

2.2 Gradient



$$f(x, y) = x^2 + y^2$$

The gradient of $f(x, y)$ is: $\nabla f = \begin{bmatrix} 2x \\ 2y \end{bmatrix}$

TASK

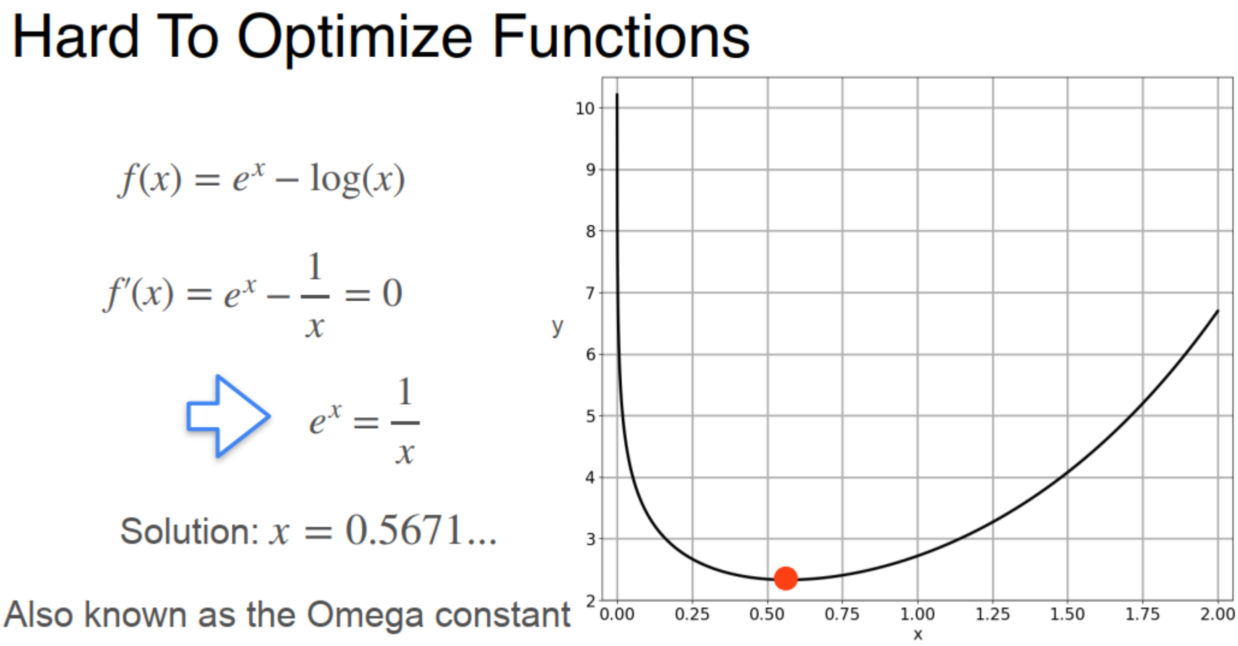
Find the gradient of $f(x, y)$ at $(2,3)$

The gradient of $f(x, y)$ is given as:

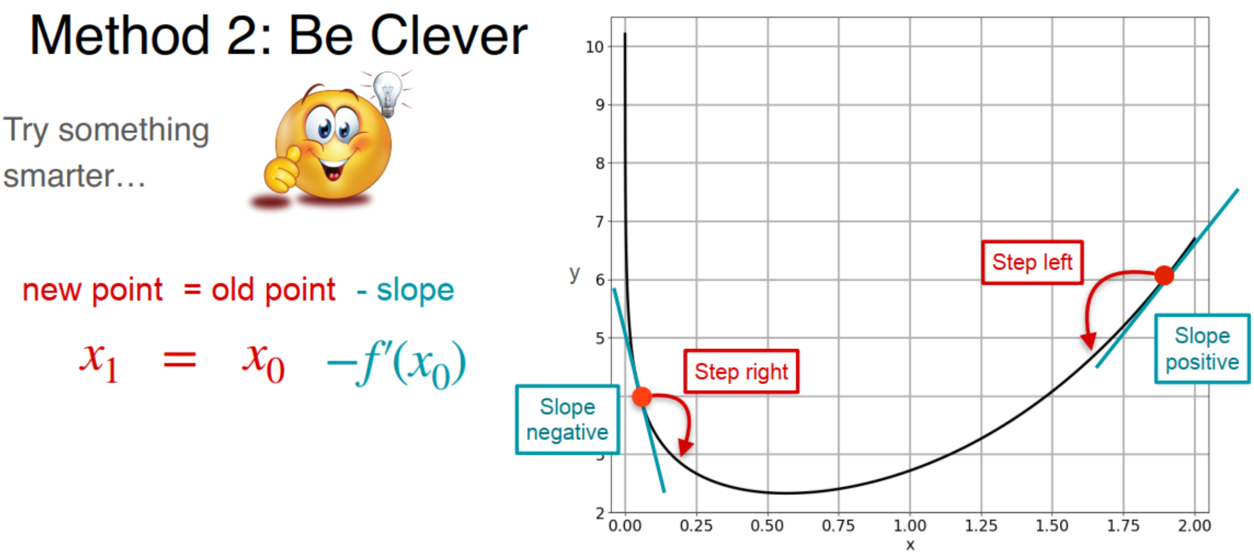
$$\nabla f = \begin{bmatrix} 2 \cdot 2 \\ 2 \cdot 3 \end{bmatrix} = \begin{bmatrix} 4 \\ 6 \end{bmatrix}$$

- 一元函数的切线，对应二元函数的切平面，如上图，二元函数在某一点处的两个切线（偏导数）构成了切平面
- 梯度就是由偏导数构成的向量▽，英文为Nabla，“奈不拉”
- 梯度的方向是函数值增长最快的方向

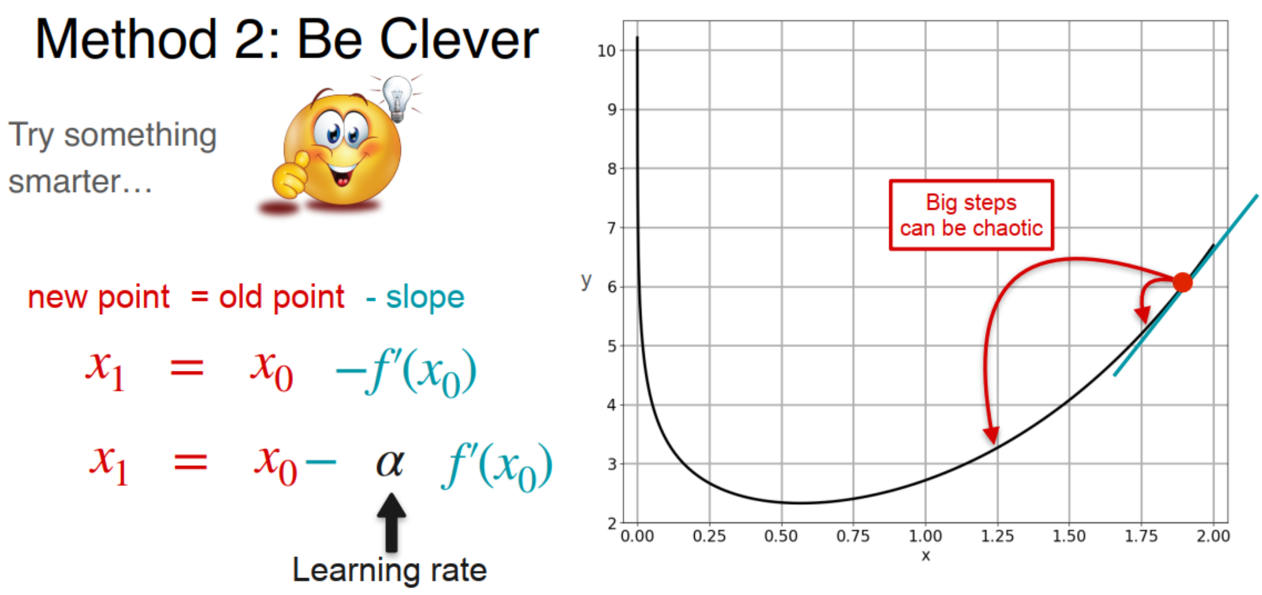
2.3 Optimization using Gradient Descent in one variable



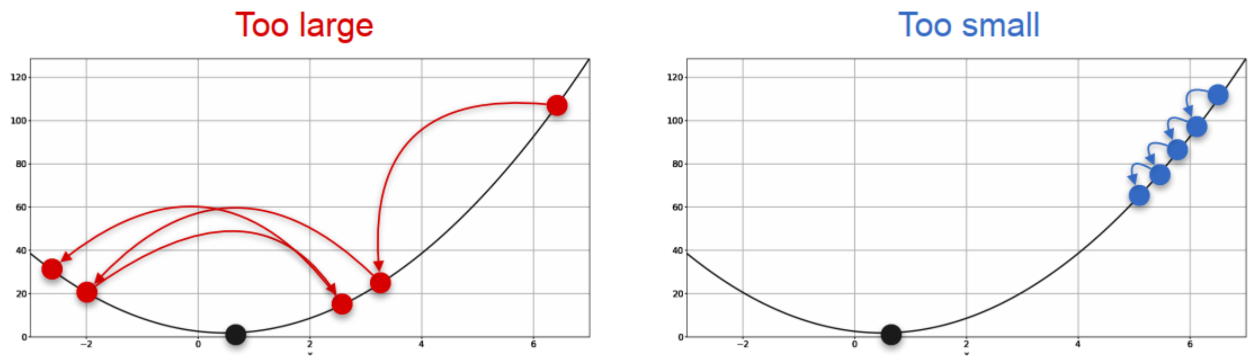
- 对于上图中的cost function，直接使用导函数=0求最小值无法实现，此时我们可以使用梯度下降法



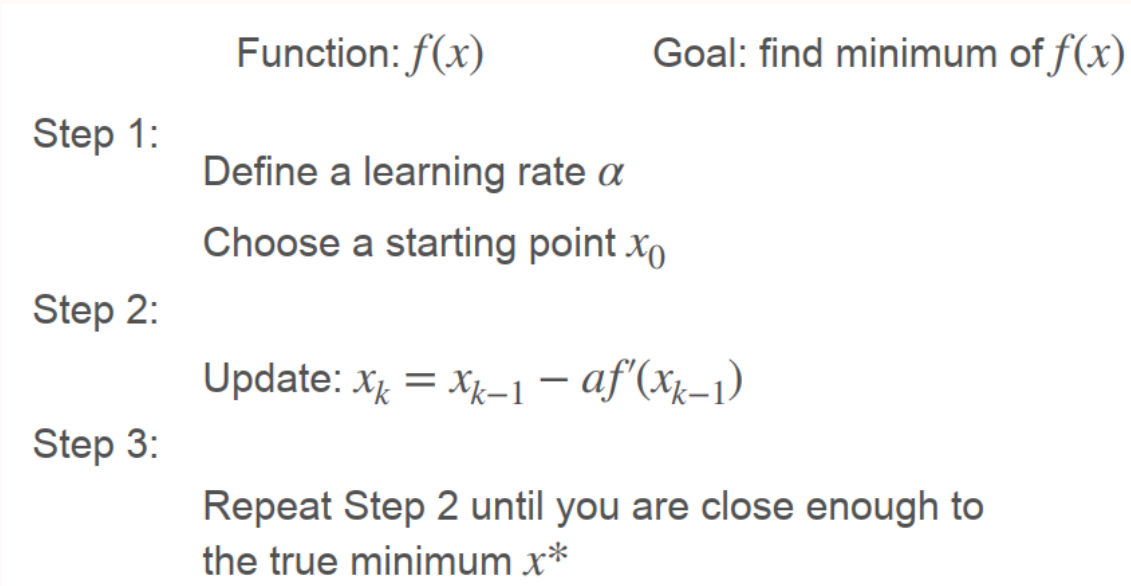
- 如上图，左边的点应该向右行走才能使函数值降低，右边的点应该向左走才能使函数值降低
- 左边点处的斜率为负数（应该向正方向行走，即：x增加），右边点处的斜率为正数（应该向负方向行走，即：x减少）
- 因此点的移动方向（x增加/减少）应该与斜率的方向相反，点移动的距离就等于该点处导数的值
- 也就是说，越陡峭的地方移动的距离越大，越平坦的地方移动的距离越小
- 我们不希望在陡峭的地方一次移动的距离过大，从而错过了最低点，因此，我们引入了学习率进行步长的调整



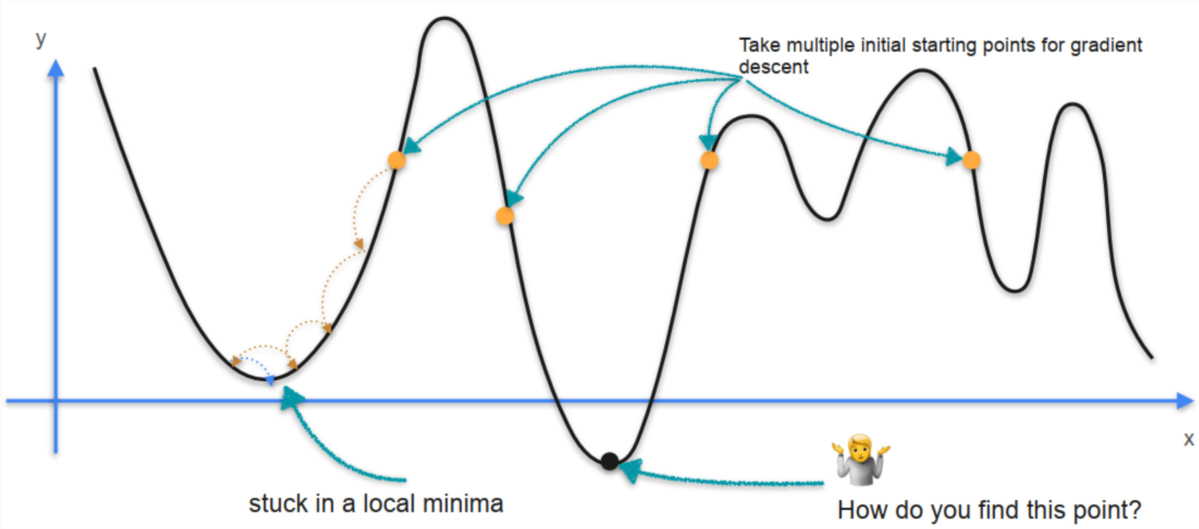
- 但是，学习率太小也不行，这会影响模型收敛的速度



🧑‍🏫对于只有一个变量的梯度下降法：



- 初始位置的选取很重要，单一的初始位置可能会使得梯度下降算法陷入局部最小值（**local minima**）



2.4 Optimization using Gradient Descent in two variables

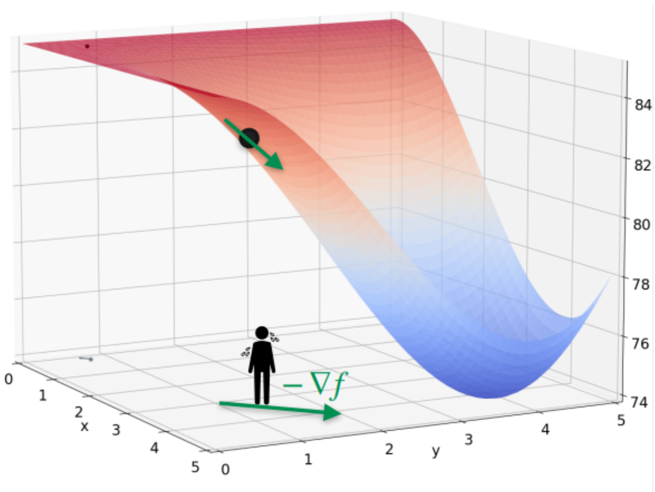
Initial position: (x_0, y_0)

Direction of greatest ascent: ∇f

Direction of greatest descent: $-\nabla f$

Updated position: $\underbrace{(x_0, y_0) - \alpha \nabla f}_{(x_1, y_1)}$

Better point!



- 沿着梯度的反方向行走~

对于含有两个变量的梯度下降法：

Function: $f(x, y)$ Goal: find minimum of $f(x, y)$

Step 1:

Define a learning rate α

Choose a starting point (x_0, y_0)

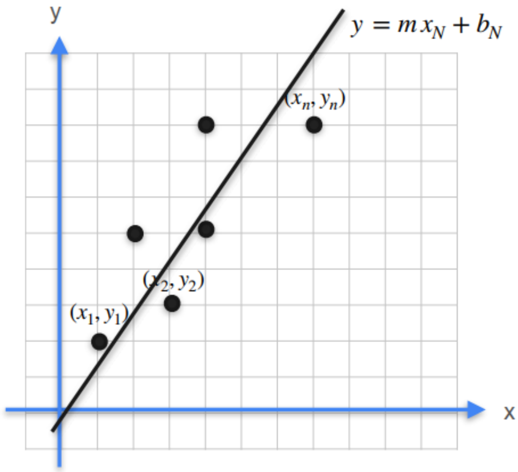
Step 2:

Update: $\begin{bmatrix} x_k \\ y_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \end{bmatrix} - \alpha \nabla f(x_{k-1}, y_{k-1})$

Step 3:

Repeat Step 2 until you are close enough to the true minimum (x^*, y^*)

2.5 梯度下降法应用于线性回归



$$\mathscr{L}(m,b) = \frac{1}{2m}\left[(mx_1 + b - y_1)^2 + \cdots + (mx_n - y_n)^2\right]$$

$$\begin{bmatrix} m_N \\ b_N \end{bmatrix} \rightarrow \begin{bmatrix} m_N \\ b_N \end{bmatrix} = \begin{bmatrix} m_{N-1} \\ b_{N-1} \end{bmatrix} - \alpha \nabla \mathscr{L}_1(m_{N-1}, b_{N-1})$$

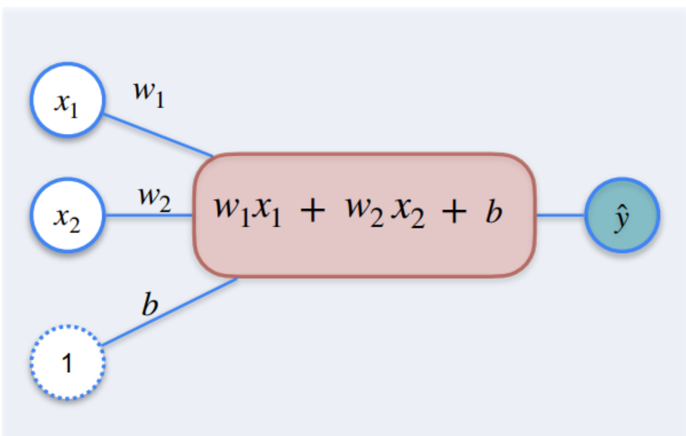
Week 3: Optimization in Neural Networks and Newtown's Method

3.1 感知机

3.1.1 分类问题

- 多元线性回归问题（损失使用平方损失）：

Single Layer Neural Network Perceptron



Prediction Function:

$$\hat{y} = w_1x_1 + w_2x_2 + b$$

Loss Function:

$$L(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$$

Main Goal:

Find w_1 , w_2 , b that give $\hat{y}$ with the least error

- 反向传播计算梯度：

Prediction Function:

$$\hat{y} = w_1x_1 + w_2x_2 + b$$

Loss Function:

$$L(y, \hat{y}) = \frac{1}{2}(y - \hat{y})^2$$

Using chain rule:

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial b}$$

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_1}$$

$$\frac{\partial L}{\partial w_2} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w_2}$$

- 计算结果：

Main Goal:

Find w_1 , w_2 , b that give $\hat{y}$ with the least error

ie. optimal values for:
w_1 , w_2 , b

Perform Gradient Descent

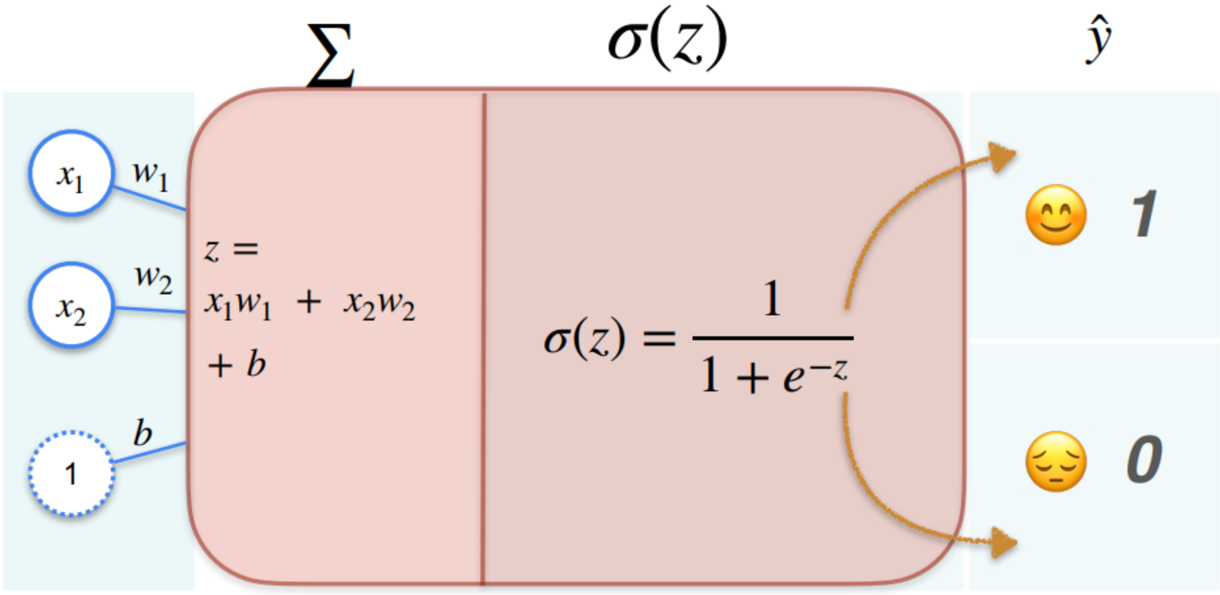
$$w_1 = w_1 - \alpha(-x_1(y - \hat{y}))$$

$$w_2 = w_2 - \alpha(-x_2(y - \hat{y}))$$

$$b = b - \alpha(-(y - \hat{y}))$$

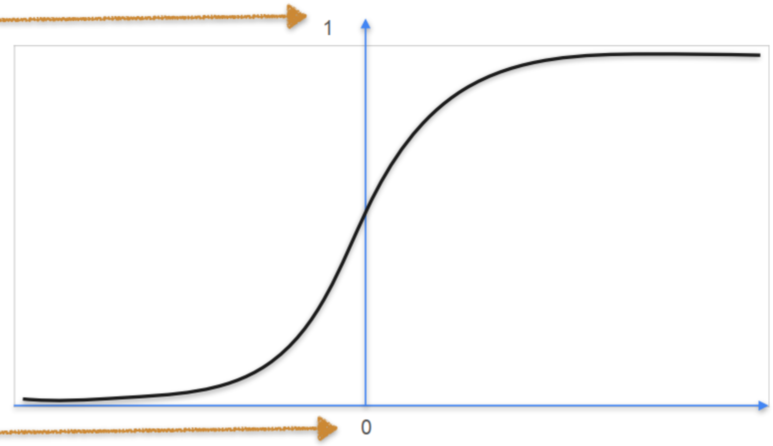
3.1.2 分类问题

- 二分类问题（激活函数使用sigmoid，损失使用对数损失）：



- 关于sigmoid函数：
  - 函数图像：

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

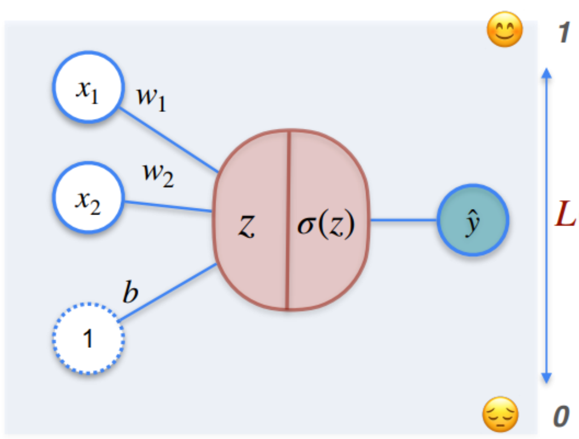


  - 导函数：

Recall that: $\sigma(z) = \frac{1}{1+e^{-z}}$

$$\frac{d}{dz}\sigma(z) = \sigma(z)\,(1-\sigma(z))$$

- 关于对数损失：



Prediction Function:

$\hat{y} = \sigma(w_1 x_1 + w_2 x_2 + b)$

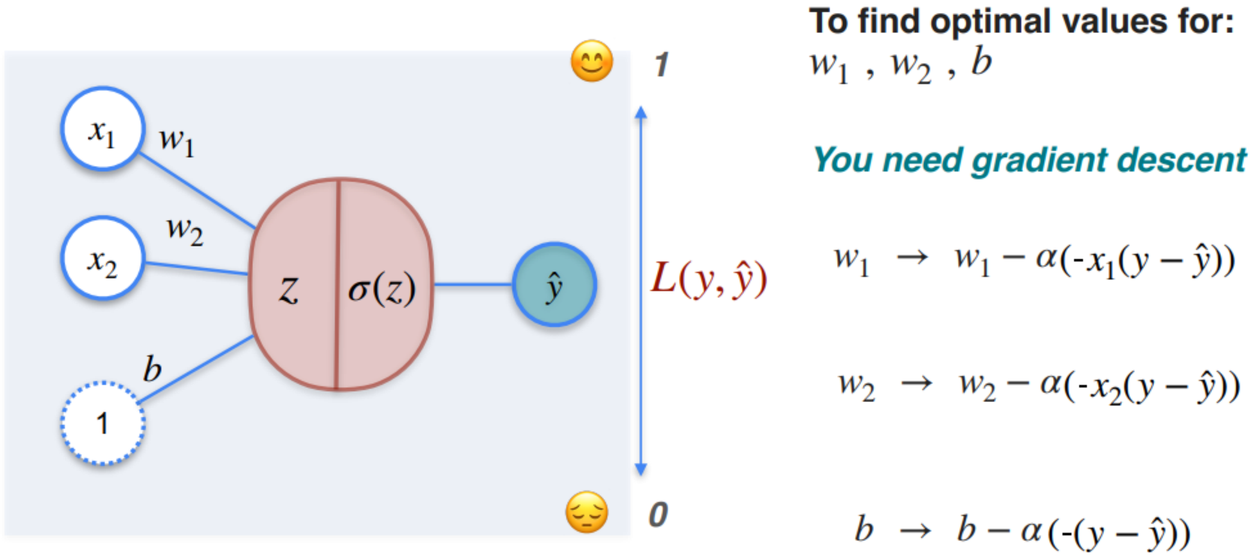
Loss Function:

$L(y, \hat{y})$ $L(y, \hat{y}) = -y\ln(\hat{y}) - (1-y)\ln(1-\hat{y})$

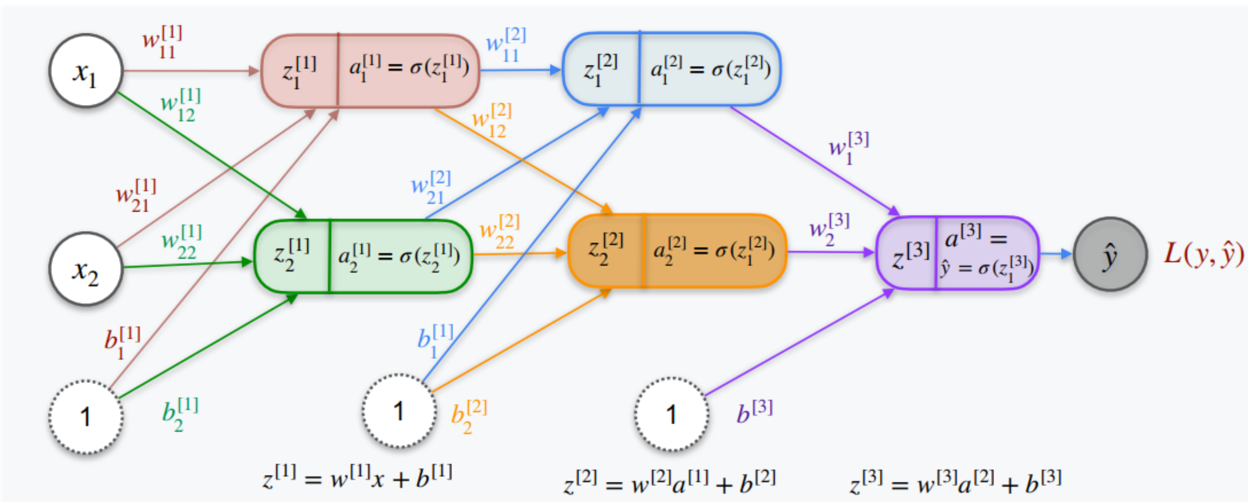
Main Goal:

Find w_1 , w_2 , b that give $\hat{y}$ with the least error

- 反向传播计算结果：

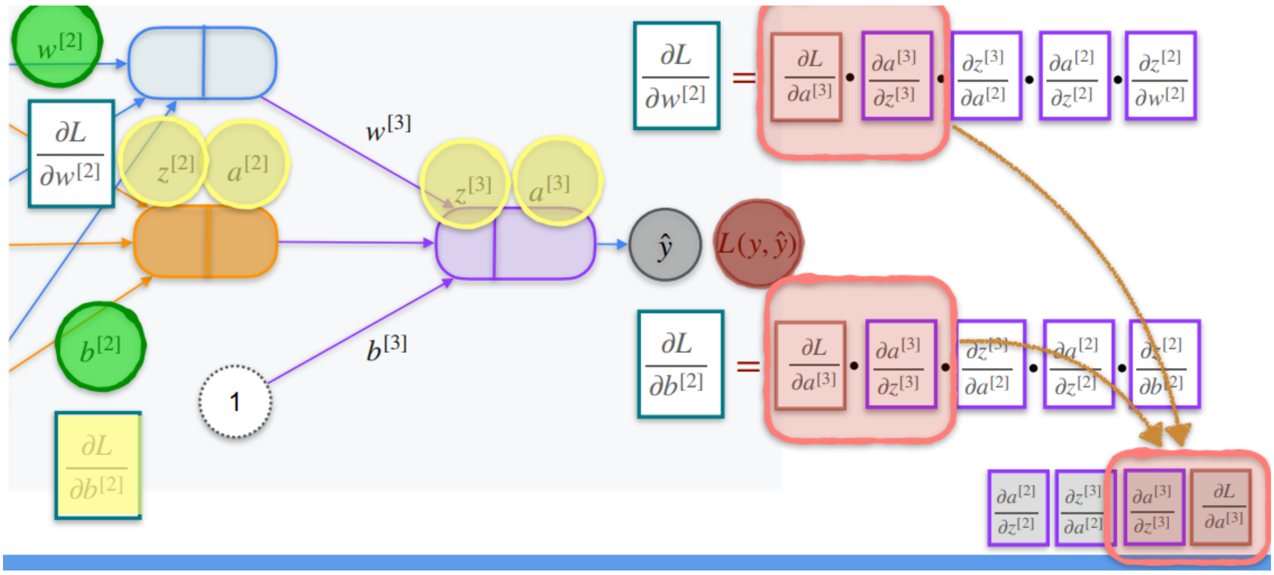


3.2 神经网络



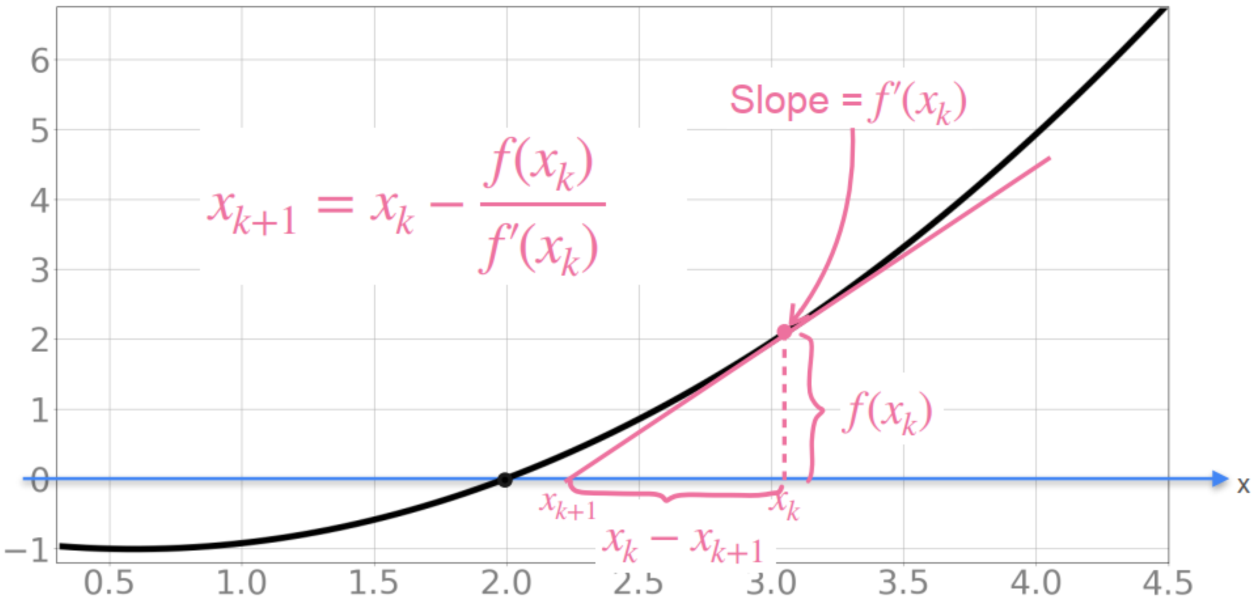
$L(y,\hat{y}) \;=\; \text{-}ylog(\hat{y}) \;-\; (1-y)log(1-\hat{y})$

- 反向传播：



3.3 牛顿方法：Newton's method

牛顿方法原本是用来找函数零点的！！！



- 牛顿方法用于优化：

Newton’s Method for Optimization

Newton’s method

Goal: find a zero of $f(x)$

1) Start with some x_0

2) Update:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

3) Repeat 2) until you find the root.

NM for Optimization

Goal: minimize $g(x)$ ⇒ find zeros of $g'(x)$

$f(x) \mapsto g'(x) \qquad f'(x) \mapsto (g'(x))'$

1) Start with some x_0

2) Update:

$$x_{k+1} = x_k - \frac{g'(x_k)}{(g'(x_k))'}$$

3) Repeat 2) until you find the root.

> 找到函数g(x)的最小值点，相当于找g'(x)的零点，这样以来就把优化问题转换成了找零点问题，牛顿方法便可派上用场！

3.4 二阶导数

- 表示方法（一元函数）：

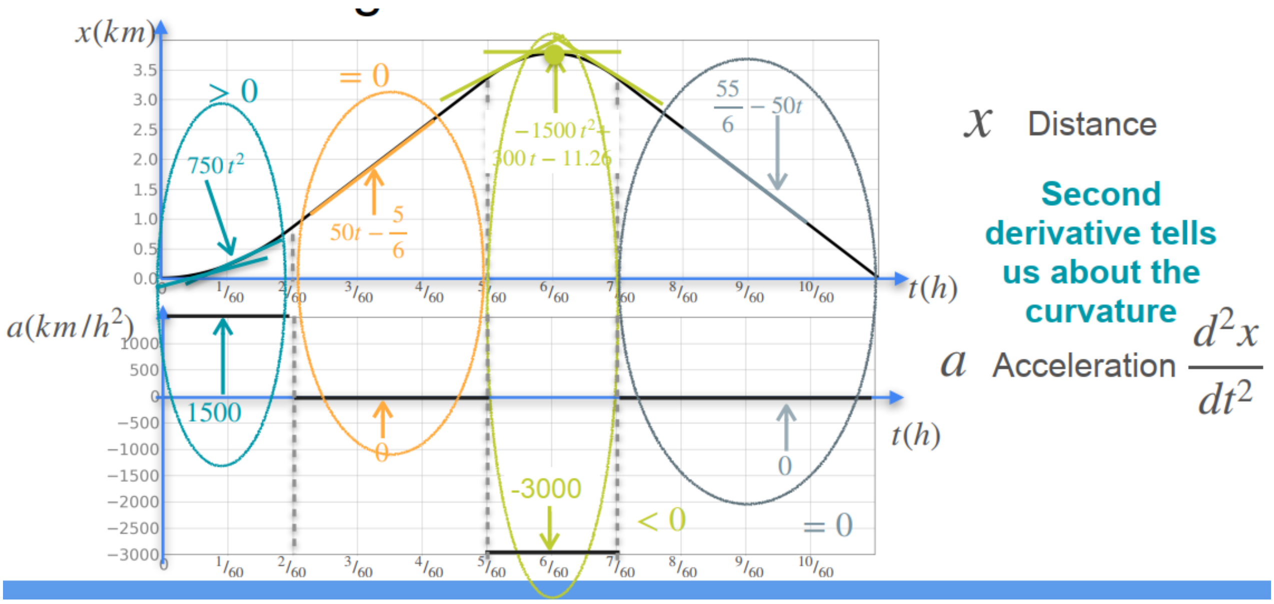
Leibniz notation: $\dfrac{d^2f(x)}{dx^2} = \dfrac{d}{dx}\left(\dfrac{df(x)}{dx}\right)$

Lagrange notation: $f''(x)$

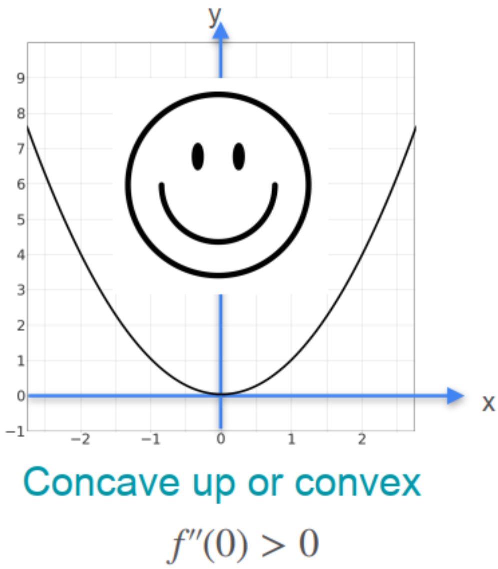
- 表示方法（二元函数）：

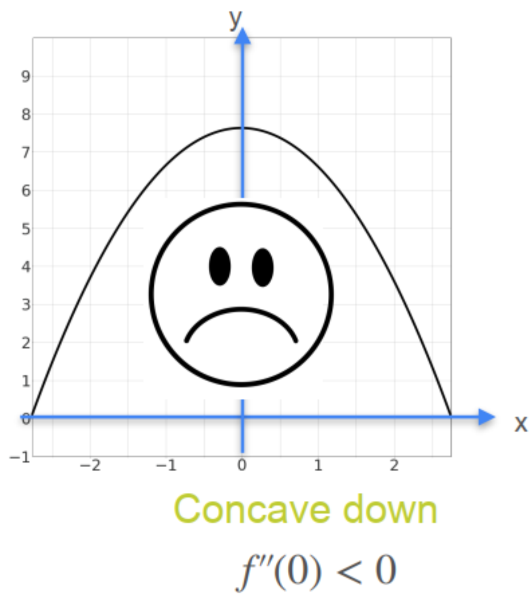
	Leibniz's notation	Lagrange's notation
Rate of change of $f'_x(x, y)$ w.r.t x	$\dfrac{\partial^2 f}{\partial x^2}$	$f_{xx}(x, y)$
Rate of change of $f'_y(x, y)$ w.r.t y	$\dfrac{\partial^2 f}{\partial y^2}$	$f_{yy}(x, y)$
Rate of change of $f'_x(x, y)$ w.r.t y	$\dfrac{\partial^2 f}{\partial x \partial y}$	$f_{xy}(x, y)$
Rate of change of $f'_y(x, y)$ w.r.t x	$\dfrac{\partial^2 f}{\partial y \partial x}$	$f_{yx}(x, y)$

- 😇含义：二阶导数可以用于衡量曲线和直线的偏离量，即：曲率



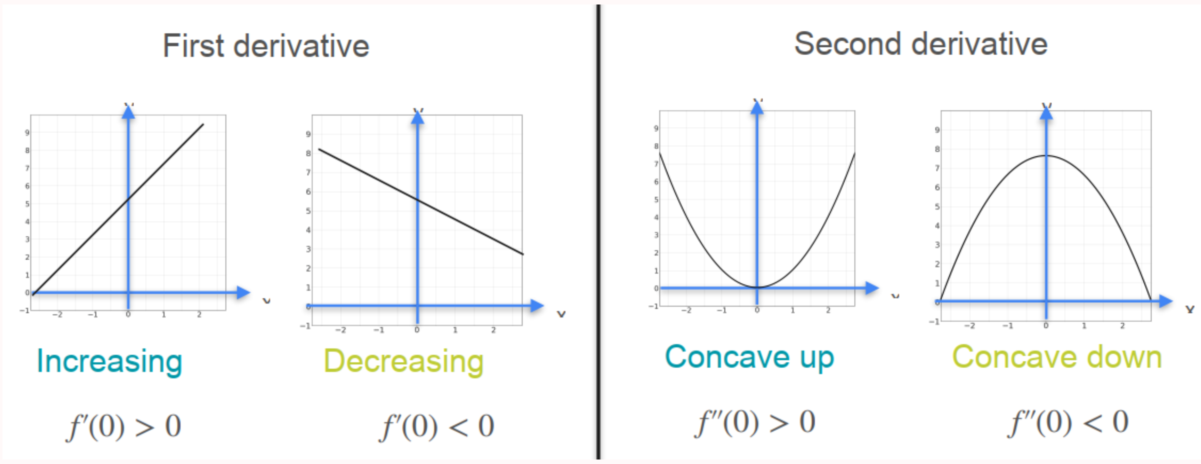
- 二阶导数>0为凹函数，二阶导数<0为凸函数



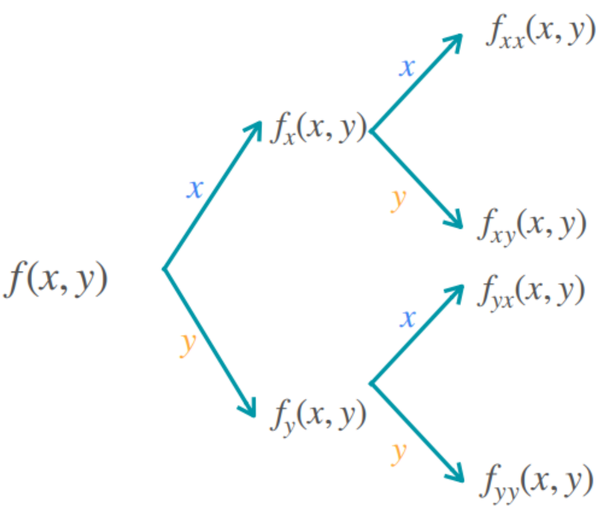


当我们找到了一些使得一阶导数等于0的点，我们并不能确定这些点是局部最大值还是局部最小值，这时候我们可以计算该点处的二阶导数，如果二阶导数大于0，则为局部最小值点，反之为局部最大值点！

- 一阶导数反映单调性，二阶导数反映凹凸性



3.5 海森矩阵：Hessian Matrix



$$\nabla f_x = \begin{bmatrix} f_{xx}(x,y) \\ f_{xy}(x,y) \end{bmatrix} \qquad \nabla f_y' = \begin{bmatrix} f_{yx}(x,y) \\ f_{yy}(x,y) \end{bmatrix}$$

$$H = \begin{bmatrix} f_{xx}(x,y) & f_{xy}(x,y) \\ f_{yx}(x,y) & f_{yy}(x,y) \end{bmatrix} = \begin{bmatrix} \nabla f_x^T \\ \nabla f_y^T \end{bmatrix}$$

Hessian matrix

All information about second derivatives

一元函数和二元函数对比：

	1 variable		2 variables
Function	$f(x)$		$f(x,y)$
First derivative	$f'(x)$	Rate of change of $f(x)$	$f_x(x,y)$ Rate of change w.r.t x $f_y(x,y)$ Rate of change w.r.t y $\nabla f = \begin{bmatrix} f_x(x,y) \\ f_y(x,y) \end{bmatrix}$
Second derivative	$f''(x)$	Rate of change of the rate of change of $f(x)$	$H(x,y) = \begin{bmatrix} f_{xx}(x,y) & f_{xy}(x,y) \\ f_{yx}(x,y) & f_{yy}(x,y) \end{bmatrix}$

3.6 海森矩阵和凹凸性

	1 variable $f(x)$	2 variables $f(x,y)$	More variables $f(x_1, x_2, \ldots, x_n)$
(Local) minima	Happy face $f''(x) > 0$	Upper paraboloid $\lambda_1 > 0 \,\&\, \lambda_2 > 0$	All $\lambda_i > 0$
(Local) maxima	Sad face $f''(x) < 0$	Down paraboloid $\lambda_1 < 0 \,\&\, \lambda_2 < 0$	All $\lambda_i < 0$
Need more information	$f''(x) = 0$	Saddle point $\lambda_1 > 0 \,\&\, \lambda_2 < 0$ $\lambda_1 < 0 \,\&\, \lambda_2 > 0$ Or some $\lambda_i = 0$	Some $\lambda_i > 0$ and some $\lambda_j < 0$ OR At least one $\lambda_i = 0$

3.7 牛顿方法用于两个变量的函数

1 variable

$$x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$$

$$x_{k+1} = x_k - \boxed{f''(x_k)^{-1}}\boxed{f'(x_k)}$$

2 variables

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} - H^{-1}(x_k, y_k)\ \nabla f(x_k, y_k)$$

> 注意：Hessian矩阵和梯度向量的乘积顺序，海森矩阵形状为2x2，梯度向量形状为2x1