1 机器学习算法概述

0 机器学习的发展

- **机器学习的发展史**
  - 机器学习是人工智能应用研究较为重要的分支，它的发展过程大体上可分为4个时期：

    1.第一阶段是在50年代中叶到60年代中叶，属于热烈时期。

    2.第二阶段在60年代中叶至70年代中叶，被称为机器学习的冷静时期。

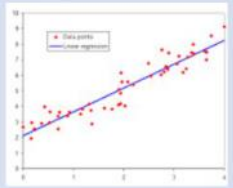
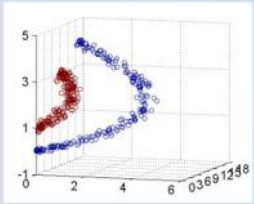
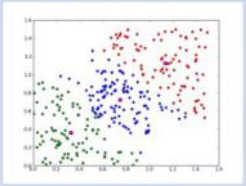
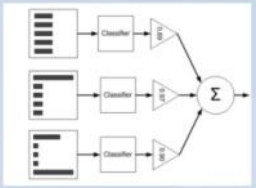
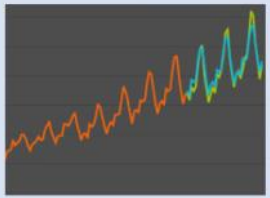
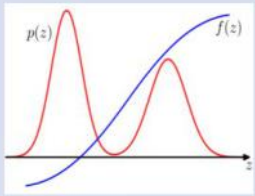
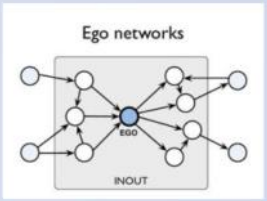
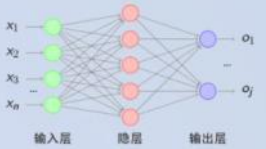
    3.第三阶段从70年代中叶至80年代中叶，称为复兴时期。

    4.机器学习的最新阶段始于1986年。一方面，由于神经网络研究的重新兴起，另一方面，对实验研究和应用研究得到前所未有的重视。我国的机器学习研究开始进入稳步发展和逐渐繁荣的新时期。

0 机器学习的学习任务分类

- 监督式学习
- 无监督学习
- 强化学习

0 机器学习经典算法

回归	分类	聚类	集成
			
时间序列	**近似推断**	**图算法**	**神经网络**
			

0 机器学习入门

对一个领域进行学习的第一步就是尽快的了解全貌搭建出整体的知识体系，然后在实践中不断提升对该领域的认识。

- **1．数学知识**
- **2．编程语言**
- **3．经典机器学习理论和基本算法**
- **4．动手实践机器学习**

1.1 什么是机器学习

计算机通过模拟或实现人类的学习行为，以获取新的知识或技能，重新组织已有的知识结构使之达到不断改善自身智能的目的。

- **机器学习(Machine Learning)**：计算机能模拟人的学习行为，自动地通过学习获取知识和技能，不断改善性能，实现自我完善。

1）**学习机理**：对学习机制的研究，即人类获取知识、技能和抽象概念的天赋能力。

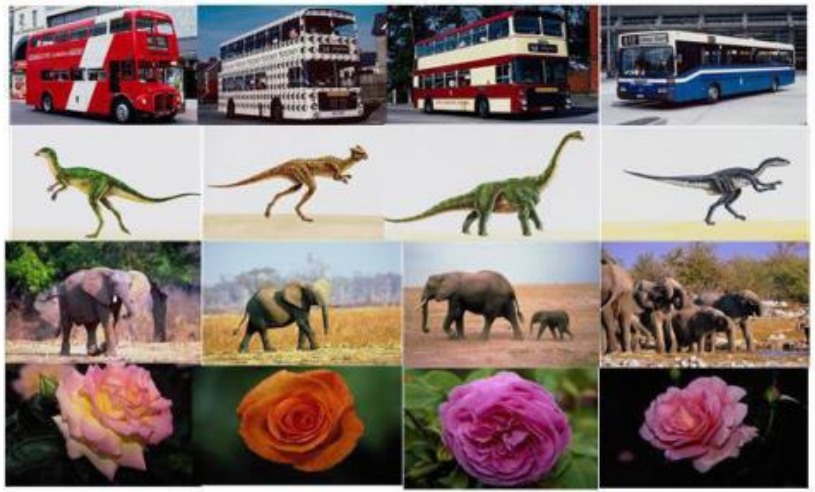
2）**学习方法**：在生物学习机理进行简化的基础上，用计算的方法进行再现。

3）**学习系统**：根据特定任务的要求，建立相应的学习系统。

机器学习的定义：

给定一个任务 T，一些相关的经验 E，以及关于学习效果的一个度量 P，机器学习就是通过对经验 E 的学习，来优化度量 P 的一个过程。

机器学习具体干什么事儿？

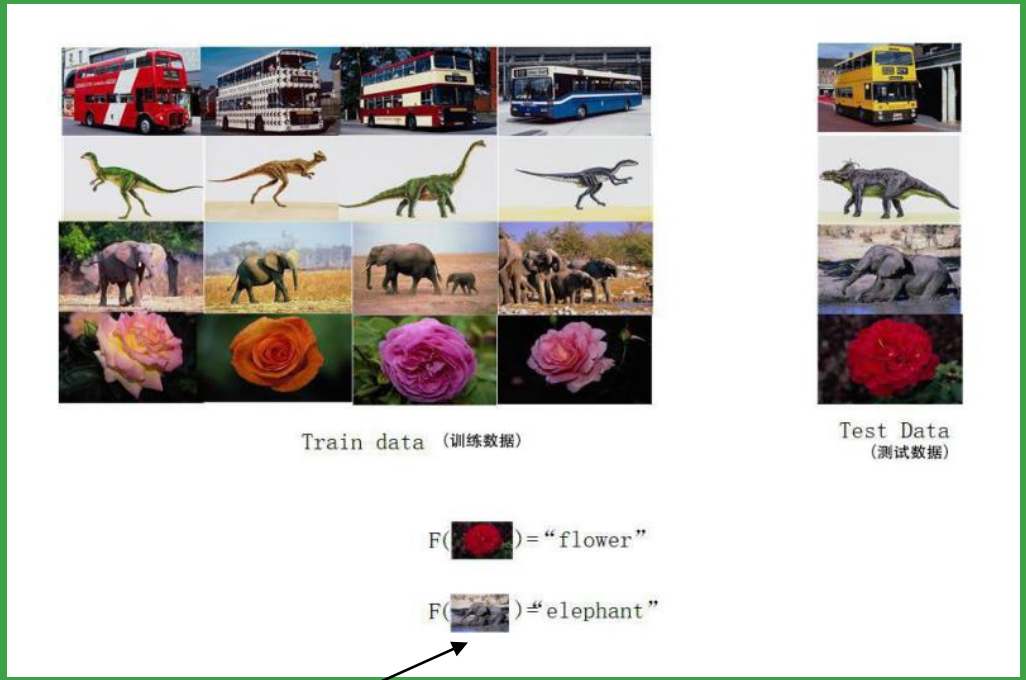


Train data （训练数据）

Test Data
（测试数据）

F()=“flower”

F()=“elephant”



回顾下高中学的函数知识:

y=f(x)

其中的F()函数，就相当于将一张图片映射到标签空间上。

y=f(x)当中，y和x已知，如何构造合理的映射关系？

Or 如何贴近真实情况？

介绍方法：

1.损失函数

2.误差反向传播

损失函数：

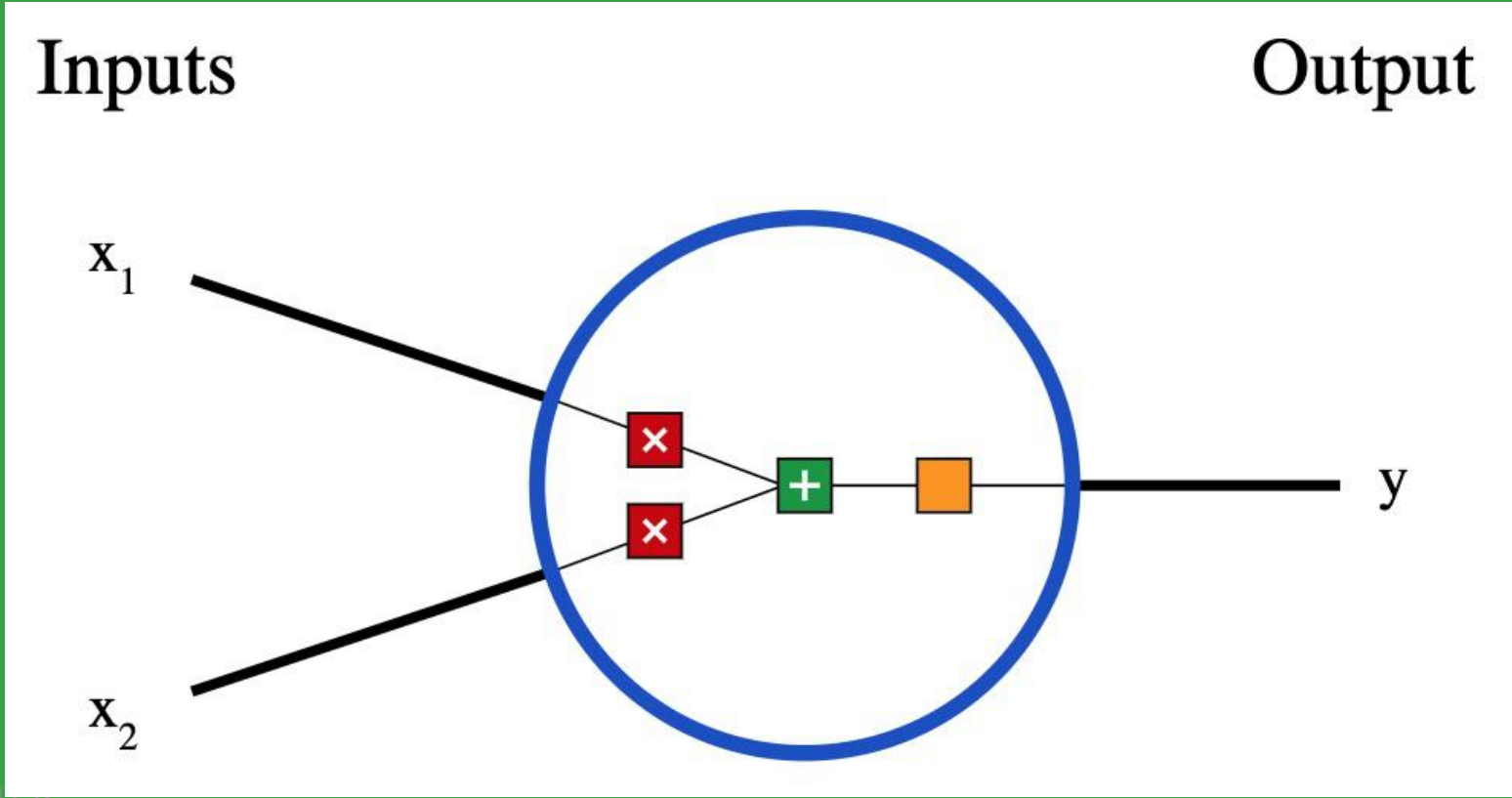
其中评价模型学习效果好不好的指标，就是模型的损失！！！！！

具体形式为a = loss_function(Y_true,Y_predect)

Loss_funtion有很多，
常见的MSE，MAE

a = loss_function(Y_true,F(X_true))

误差反向传播算法（Loss Back Propagation, BP）



误差反向传播算法（Loss Back Propagation, BP）

类似于牛顿下降法，将误差求导后反向传播更新模型中的参数，用梯度下降进行修改模型的参数。

而后再经过下一次的传播不断更新参数，使得模型参数贴近于真实情况。

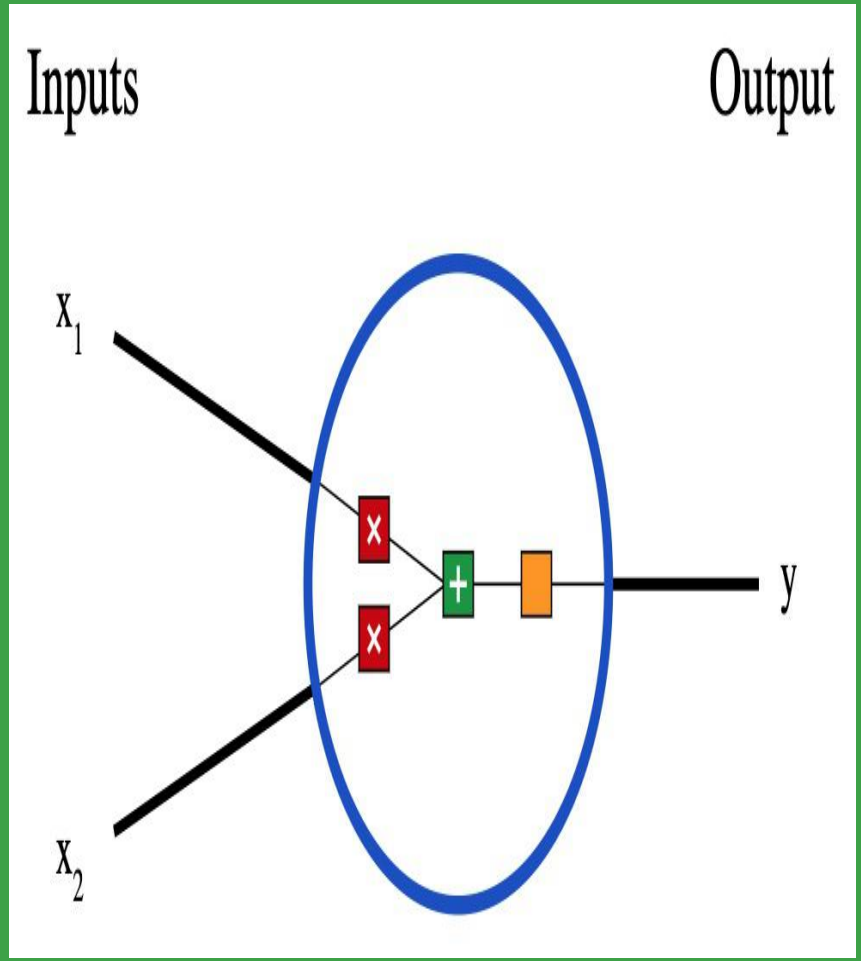
分为以下几步：

1.向前传播　2.梯度下降　3.误差反向传播

1.向前传播

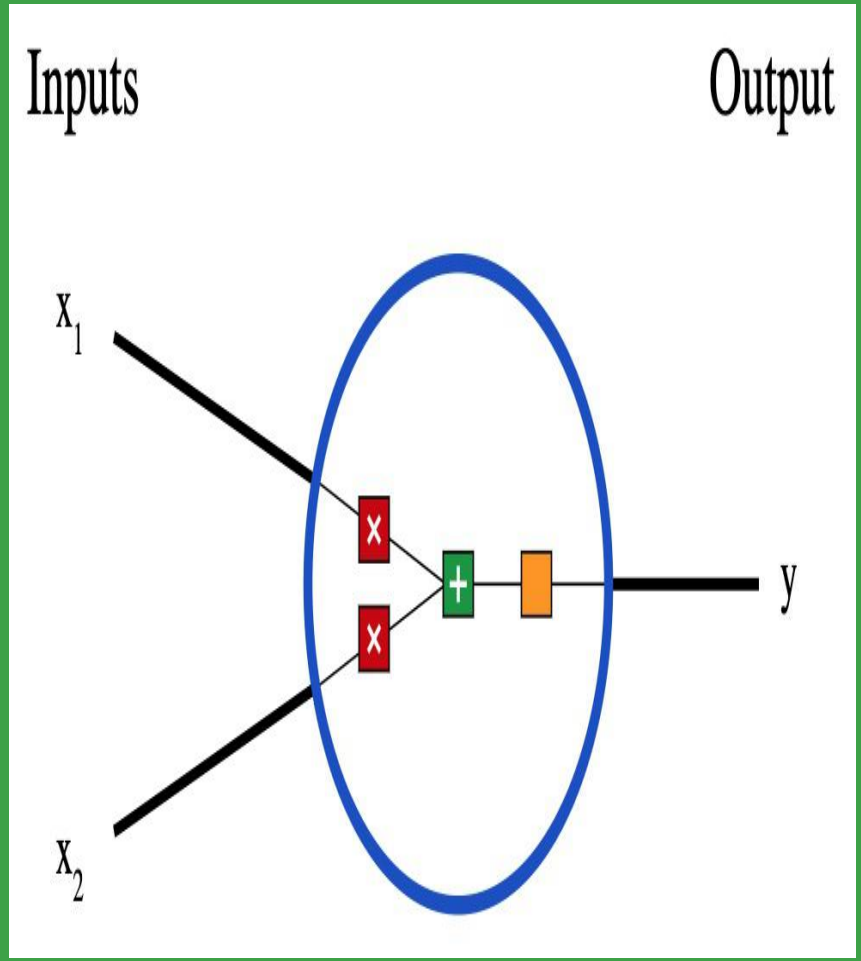
第一步：每个输入都跟一个权重相乘（红色）

$$x_1 \leftarrow x_1 \times w_1$$
$$x_2 \leftarrow x_2 \times w_2$$



第二步：加权后的输入求和，加上一个偏差b（绿色）

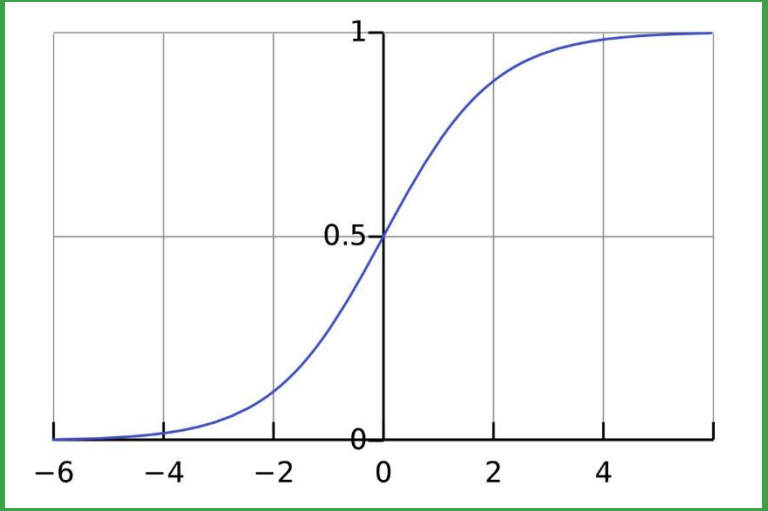
$$(x_1 \times w_1) + (x_2 \times w_2) + b$$



第三步：这个结果传递给一个激活函数f(x)（一般用Sigmoid函数）

$$y = f(x_1 \times w_1 + x_2 \times w_2 + b)$$

$$f(x) = \frac{1}{1 + e^{-x}}$$



激活函数的作用是将一个无边界的输入，转变成一个可预测的形式。

那么对f(x)求导之后f`(x) = f(x)(1 - f(x)),所以我在这里令Sigmoid求导后的函数为：deriv_sigmoid(x)

假设我们有一个神经元，激活函数就是S型函数，其参数如下：

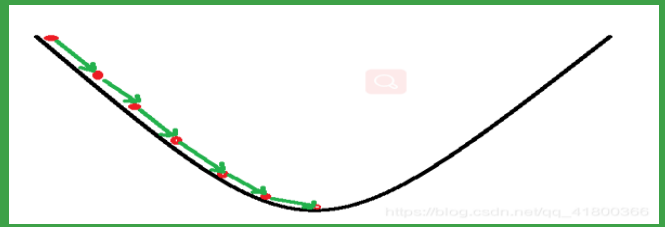
$$w = [0, 1]$$
$$b = 4$$

w=[0, 1] 就是以向量的形式表示w_1=0, w_2=1。现在，我们给这个神经元一个输入x=[2, 3]。我们用点积来表示：

$$\begin{aligned}(w \cdot x) + b &= (w_1 * x_1 + w_2 * x_2) + b \\ &= 0 * 2 + 1 * 3 + 4 \\ &= 7\end{aligned}$$

$$y = f(w \cdot x + b) = f(7) = 0.999$$

2.梯度下降



假设这样一个场景：一个人被困在山上，需要从山上下来(找到山的最低点，也就是山谷)。但此时山上的浓雾很大，导致可视度很低；因此，下山的路径就无法确定，必须利用自己周围的信息一步一步地找到下山的路。这个时候，便可利用梯度下降算法来帮助自己下山。怎么做呢，首先以他当前的所处的位置为基准，寻找这个位置最陡峭的地方，然后朝着下降方向走一步，然后又继续以当前位置为基准，再找最陡峭的地方，再走直到最后到达最低处；同理上山也是如此，只是这时候就变成梯度上升算法了

我们假设有一个单变量的函数

$$J(\theta) = \theta^2$$

函数的微分，直接求导就可以得到

$$J'(\theta) = 2\theta$$

初始化，也就是起点，起点可以随意的设置，这里设置为1

$$\theta^0 = 1$$

学习率也可以随意的设置，这里设置为0.4

$$\alpha = 0.4$$

$$\Theta^1 = \Theta^0 + \alpha \nabla J(\Theta) \rightarrow evaluated at \Theta^0$$

我们开始进行梯度下降的迭代计算过程：

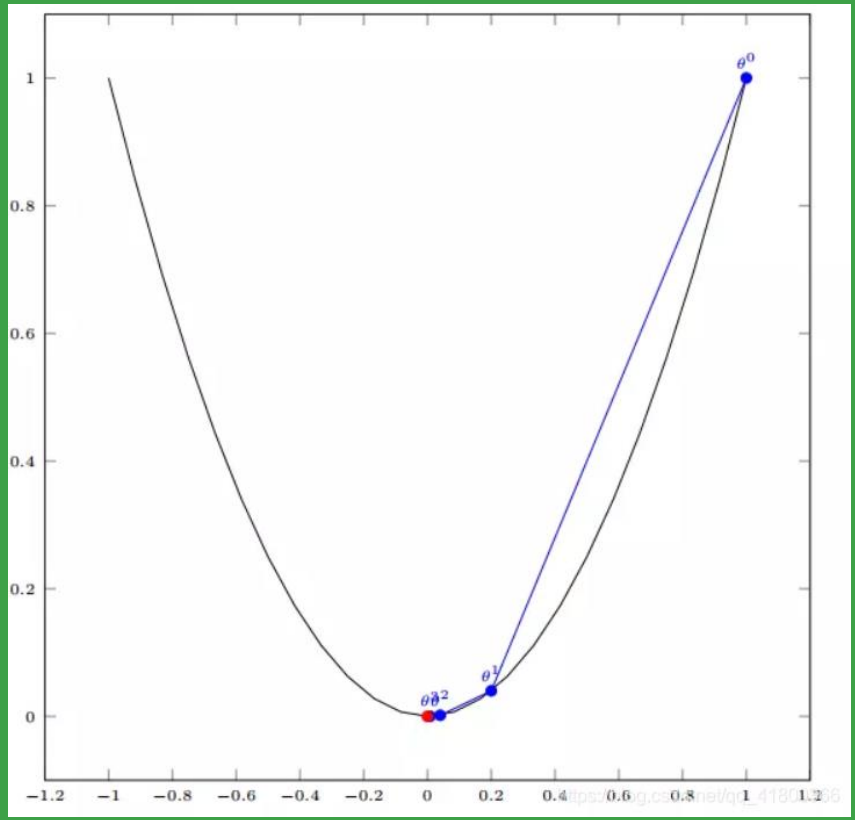
$$\theta^0 = 1$$

$$\theta^1 = \theta^0 - \alpha * J'(\theta^0) = 1 - 0.4 * 2 = 0.2$$

$$\theta^2 = \theta^1 - \alpha * J'(\theta^1) = 0.2 - 0.4 * 0.4 = 0.04$$

$$\theta^3 = 0.008$$

$$\theta^4 = 0.0016$$



如图，经过四次的运算，也就是走了四步，基本就抵达了函数的最低点，也就是山底

我们假设有一个目标函数

$$J(\Theta) = \theta_1^2 + \theta_2^2$$

现在要通过梯度下降法计算这个函数的最小值。我们通过观察就能发现最小值其实就是 (0，0)点。但是接下来，我们会从梯度下降算法开始一步步计算到这个最小值!

我们假设初始的起点为:

$$\Theta^0 = (1, 3)$$

初始的学习率为:

$$\alpha = 0.1$$

函数的梯度为:

$$\nabla J(\Theta) = \langle 2\theta_1, 2\theta_2 \rangle$$

进行多次迭代:

$$\Theta^0 = (1, 3)$$

$$\Theta^1 = \Theta^0 - \alpha \nabla J(\Theta) = (1, 3) - 0.1 * (2, 6) = (0.8, 2.4)$$

$$\Theta^2 = (0.8, 2.4) - 0.1 * (1.6, 4.8) = (0.64, 1.92)$$

$$\Theta^3 = (0.5124, 1.536)$$

$$\Theta^4 = (0.4096, 1.228800000000001)$$

$$\vdots$$

$$\Theta^{10} = (0.1073741824000003, 0.32212254720000005)$$

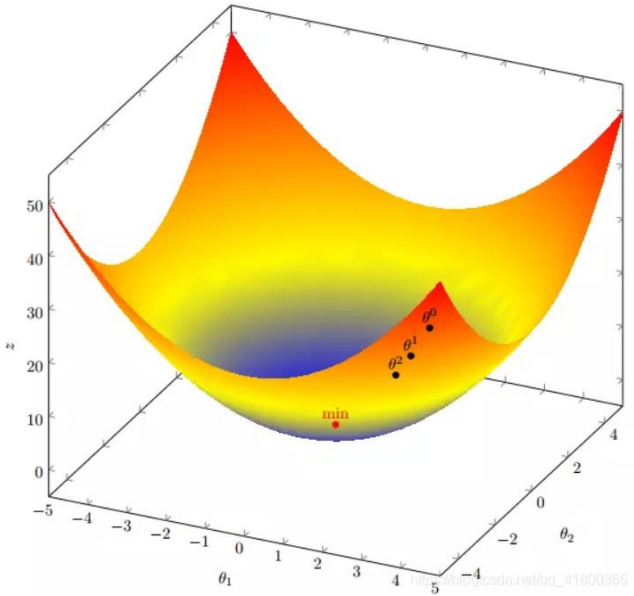
$$\vdots$$

$$\Theta^{50} = (1.141798154164342e^{-05}, 3.42539442494306e^{-05})$$

$$\vdots$$

$$\Theta^{100} = (1.6296287810675902e^{-10}, 4.8888886343202771e^{-10})$$

我们发现，已经基本靠近函数的最小值点



3.BP

假设我们有一个神经元，激活函数就是S型函数，其参数如下：

$$w = [0, 1]$$
$$b = 4$$

w=[0, 1] 就是以向量的形式表示w_1=0, w_2=1。现在，我们给这个神经元一个输入x=[2, 3]。我们用点积来表示：

$$\begin{aligned}(w \cdot x) + b &= (w_1 * x_1 + w_2 * x_2) + b \\ &= 0 * 2 + 1 * 3 + 4 \\ &= 7\end{aligned}$$

$$y = f(w \cdot x + b) = f(7) = 0.999$$

其神经元输入x=[2,3]已经明确了，若y=0.5呢？

当中会发生什么反应？

$$y = f(x_1 \times w_1 + x_2 \times w_2 + b)$$

要明确，x和y是不变的！！
误差反向传播改变的是w和b
用梯度下降改变！！

Example：

$$\Theta^1 = \Theta^0 + \alpha \nabla J(\Theta) \rightarrow evaluated at \Theta^0$$

Ours：

1.求Loss

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2$$

2.梯度下降
（通过链式法则）

$$\frac{\partial Loss}{\partial y_{pred}} = \frac{\partial (1 - y_{pred})^2}{\partial y_{pred}} = -2(1 - y_{pred})$$

Example：

$$\Theta^1 = \Theta^0 + \alpha \nabla J(\Theta) \rightarrow evaluated at \Theta^0$$

Ours：

1.求Loss

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_{true} - y_{pred})^2$$

$$\frac{\partial Loss}{\partial y_{pred}} = \frac{\partial (1 - y_{pred})^2}{\partial y_{pred}} = -2(1 - y_{pred})$$

2.梯度下降
（通过链式法则）

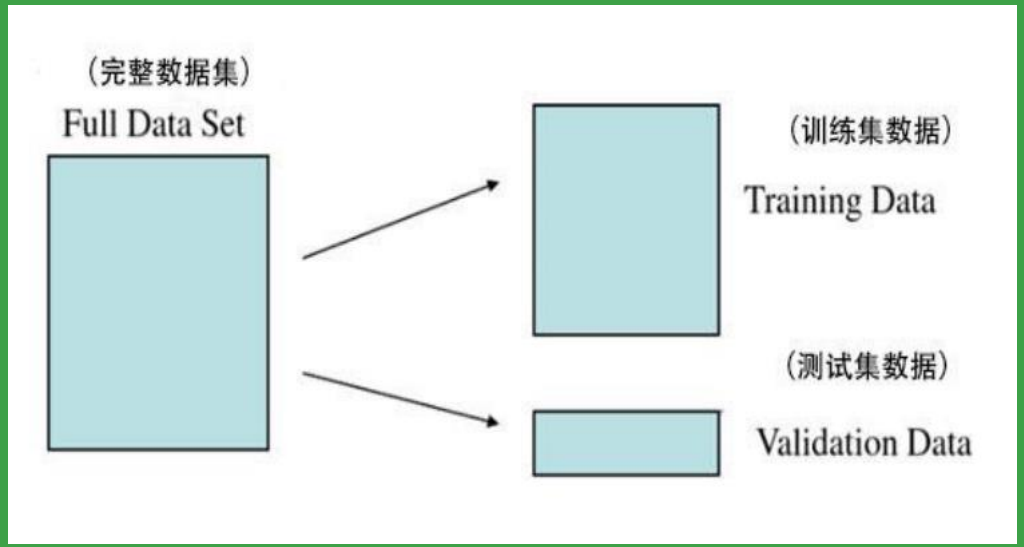
$$\frac{\partial Loss}{\partial w1} = \frac{\partial Loss}{\partial y_{pred}} \frac{\partial y_{pred}}{\partial h1} \frac{\partial h1}{\partial w1}$$

$$W_j = W_j - lr \frac{\partial Loss}{\partial W_j}$$

$$w_1 = w_1 - lr * d\_y_{pred} * d\_h1 * d\_h1\_w1$$

为何需要验证集？？

y=f(x)当中，y和x已知，如何构造合理的映射关系？



构造出映射关系之后，将新的数据带入得出的结果是否符合我们设计模型的初衷？

验证集的存在！！！