Microsoft

Reactor

一起学人工智能 - 人工智能基础

Map


Stockholm
COMING SOON
London
Redmond
Toronto
New York
San Francisco
Tel Aviv
Shanghai
Abu Dhabi
Bangalore
Sao Paulo
Sydney

个人介绍

Kinfey Lo – (卢建晖)

Microsoft Cloud Advocate

前微软MVP、Xamarin MVP和微软RD，拥有超过10年的云原生、人工智能和移动应用经验，为教育、金融和医疗提供应用解决方案。 Microsoft Iginte，Teched 会议讲师，Microsoft AI 黑客马拉松教练，目前在微软，为技术人员和不同行业宣讲技术和相关应用场景。

爱编程(Python， C#，TypeScript，Swift，Rust，Go ）

专注于人工智能，云原生，跨平台移动开发

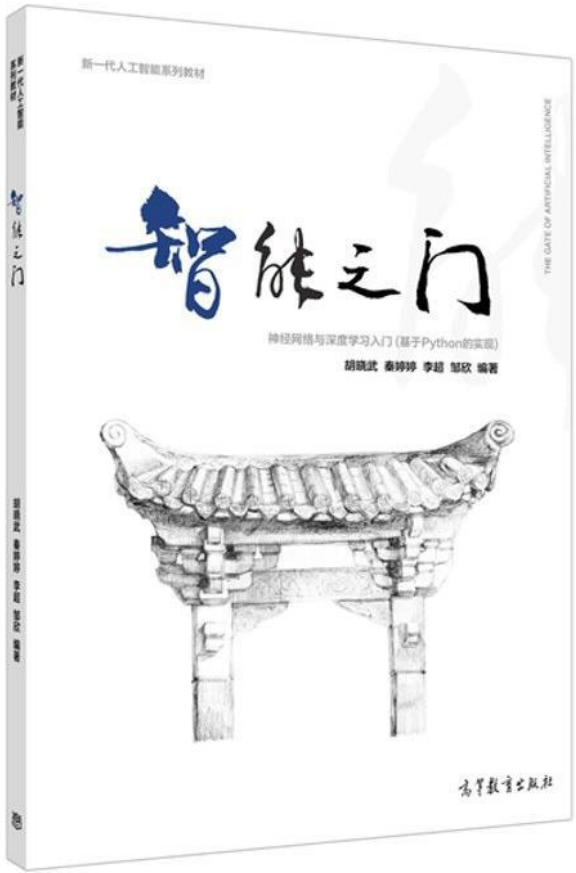
Github : https://github.com/kinfey

Email : kinfeylo@microsoft.com **Blog :** https://blog.csdn.net/kinfey

Twitter : @Ljh8304

开篇的话

一起学人工智能的资源



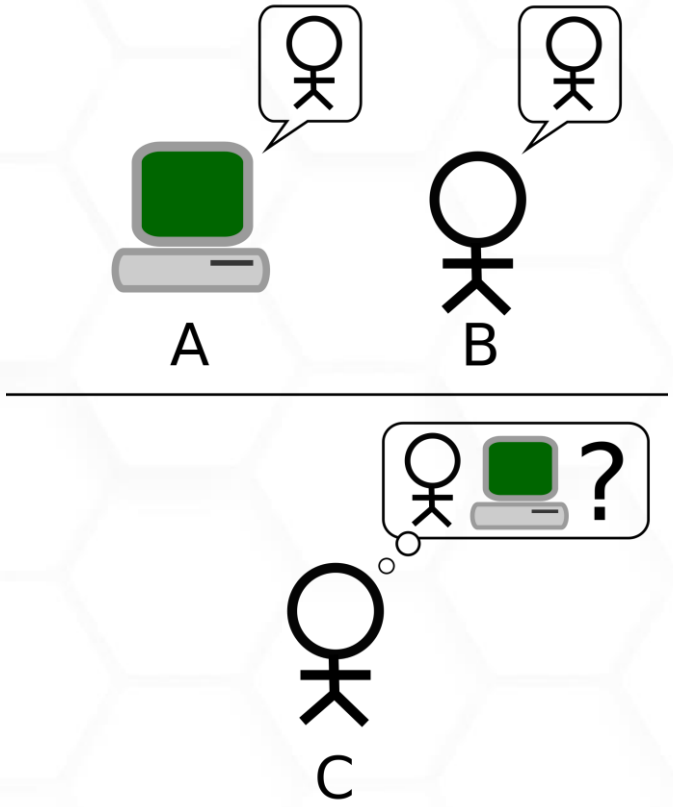
https://github.com/microsoft/ai-edu

一. 人工智能发展史

人工智能 – 图灵测试 (Turing test)

英国计算机科学家图灵于1950年提出的思想实验，目的在测试机器能否表现出与人等价或无法区分的智能。

- 自然语言
- 知识表示
- 自动推理
- 机器学习
- 计算机视觉
- 机器人学



人工智能发展


诞生
1943-1956
控制论
符号推理
高潮
1956-1974
搜索式推理
自然语言处理
反向传播算法
1970
在三到八年的时间里我们将得到一台具有人类平均智能的机器。
——马文·闵斯基
1967
一代之内……创造 ‘人工智能’ 的问题将获得实质上的解决。
——马文·闵斯基
1965
二十年内，机器将能完成人能做到的一切工作。
——艾伦·纽厄尔和赫伯特·西蒙
1958
十年之内，计算机将成为国际象棋世界冠军。
十年之内，计算机将发现并证明一个重要的数学定理。
——艾伦·纽厄尔和赫伯特·西蒙
低谷/冬天
1974-1980
运算能力不够
复杂性指数级爆炸
没有常识
“中文房间”问题
繁荣
1980-1987
专家系统五代机
神经网络
低谷/冬天
1987-1993
PC性能超过专用的AI硬件
五代机没有实用的成果
发展
1993-2006
1997深蓝
贝叶斯网络
隐马科夫模型
信息论的发展
突破
2007-
深度学习
视觉/翻译在比赛中超过人类

一个人的成长

人工智能定义

让运行程序的计算机
来学习并自动掌握某些规律

人工智能

狭义的人工智能(弱人工智能)

智能地把某件特定的事情做好，在某个领域增强人类的智慧

广义的人工智能(强人工智能)

像人类一样能认知，思考，判断：模拟人类的智能

机器学习

监督学习（Supervised Learning）

通过标注的数据来学习

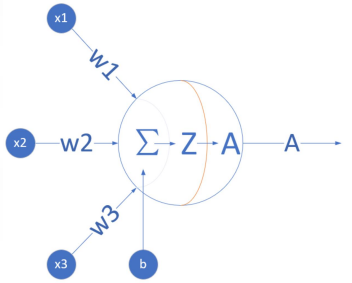
强化学习（Reinforcement Learning）

提升

无监督学习（Unsupervised Learning）

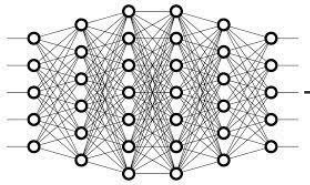
通过没有标注的数据来学习

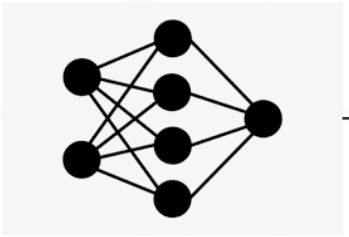
神经网络发展



M-P神经元

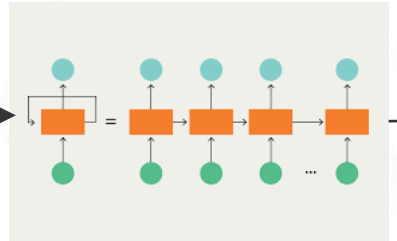
深度神经网络
DNN

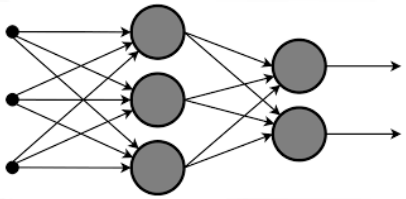




卷积深度神经网络
CNN

循环神经网络
RNN

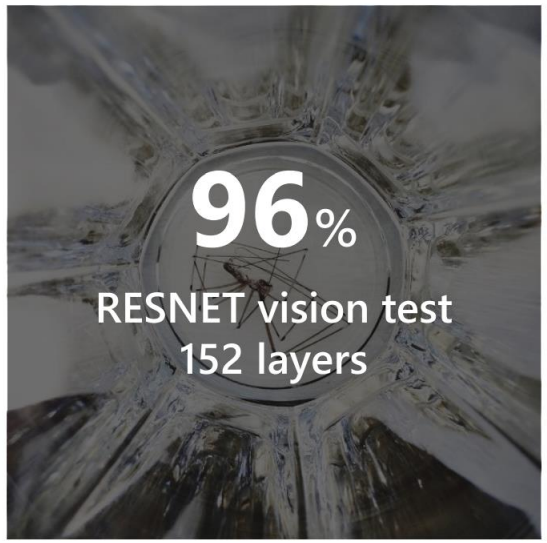




生成对抗网络
GAN

人工智能 vs 人

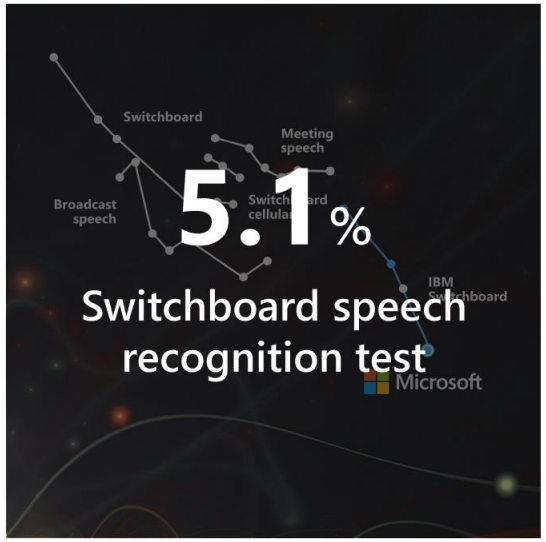
视觉
Vision



2016

Object recognition
Human parity

语音
Speech



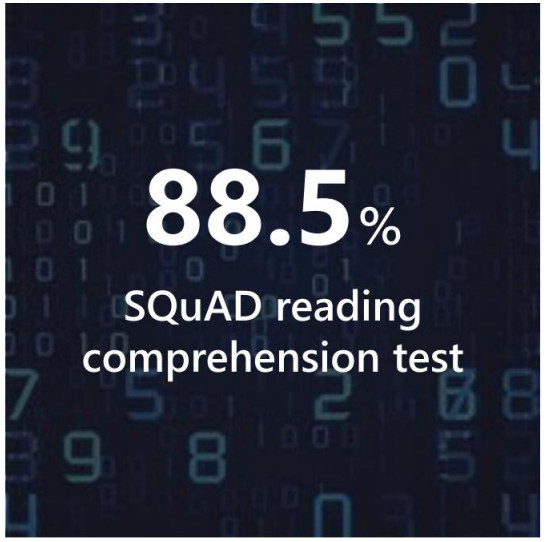
2017

Speech recognition
Human parity

March 2018

Machine translation
Human parity

语言
Language

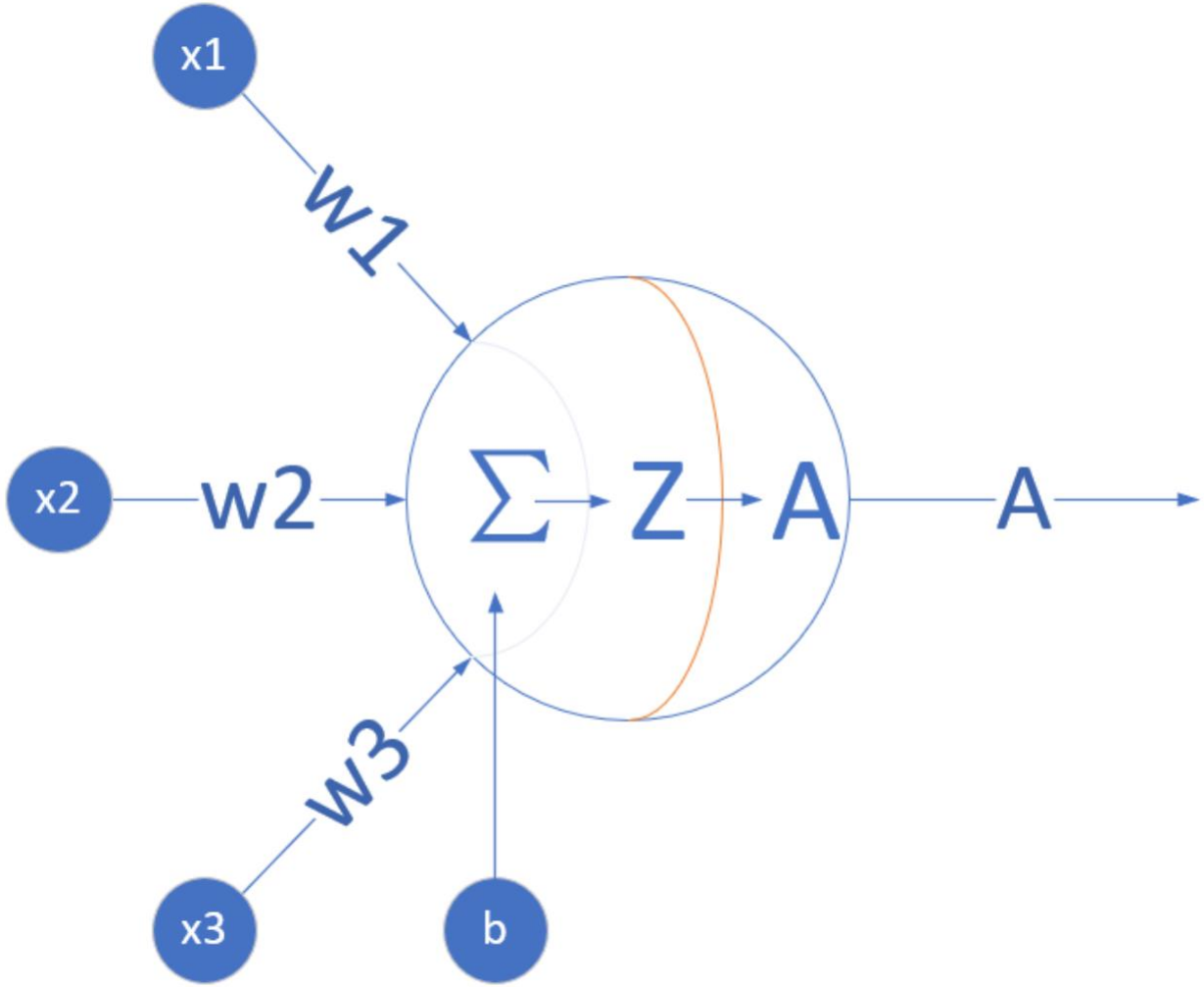


January 2018

Machine reading comprehension
Human parity

二. 神经网络的基本原理

生物学中的神经网络

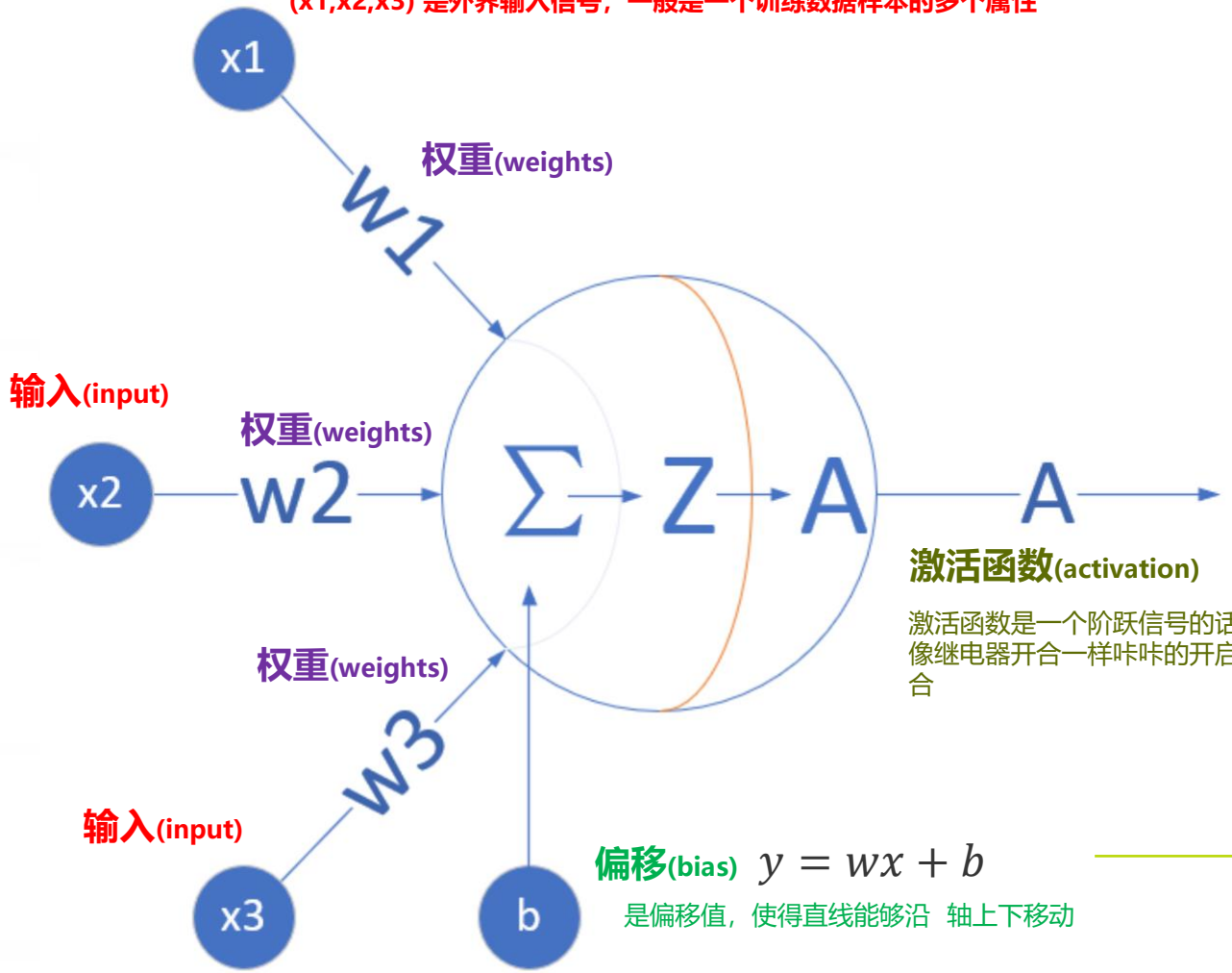


在生物神经网络中，每个神经元与其他神经元相连，当它兴奋时，就会像相邻的神经元发送化学物质，从而改变这些神经元内的电位；如果某神经元的电位超过了一个阈值，那么它就会被激活（兴奋），向其他神经元发送化学物质。把许多这样的神经元按照一定的层次结构连接起来，我们就构建了一个神经网络

计算机科学的神经网络

输入(input)

(x1,x2,x3) 是外界输入信号，一般是一个训练数据样本的多个属性



激活函数(activation)

激活函数是一个阶跃信号的话，会像继电器开合一样咔咔的开启和闭合

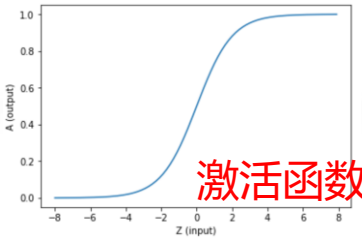
偏移(bias) $y = wx + b$

是偏移值，使得直线能够沿 轴上下移动

激活函数(activation)

求和之后，神经细胞已经处于兴奋状态了，已经决定要向下一个神经元传递信号了，但是要传递多强烈的信号，要由激活函数来确定：

$$A = \sigma(Z)$$



激活函数是有渐变过程的

偏移(bias) $= \sum_{i=1}^{m}(w_i x_i) + b$ =WX+b

在脑神经细胞中，一定是输入信号的电平/电流大于某个临界值时，神经元细胞才会处于兴奋状态，这个 b 实际就是那个临界值

$$w_1x_1 + w_2x_2 + w_3x_3 \geq t$$

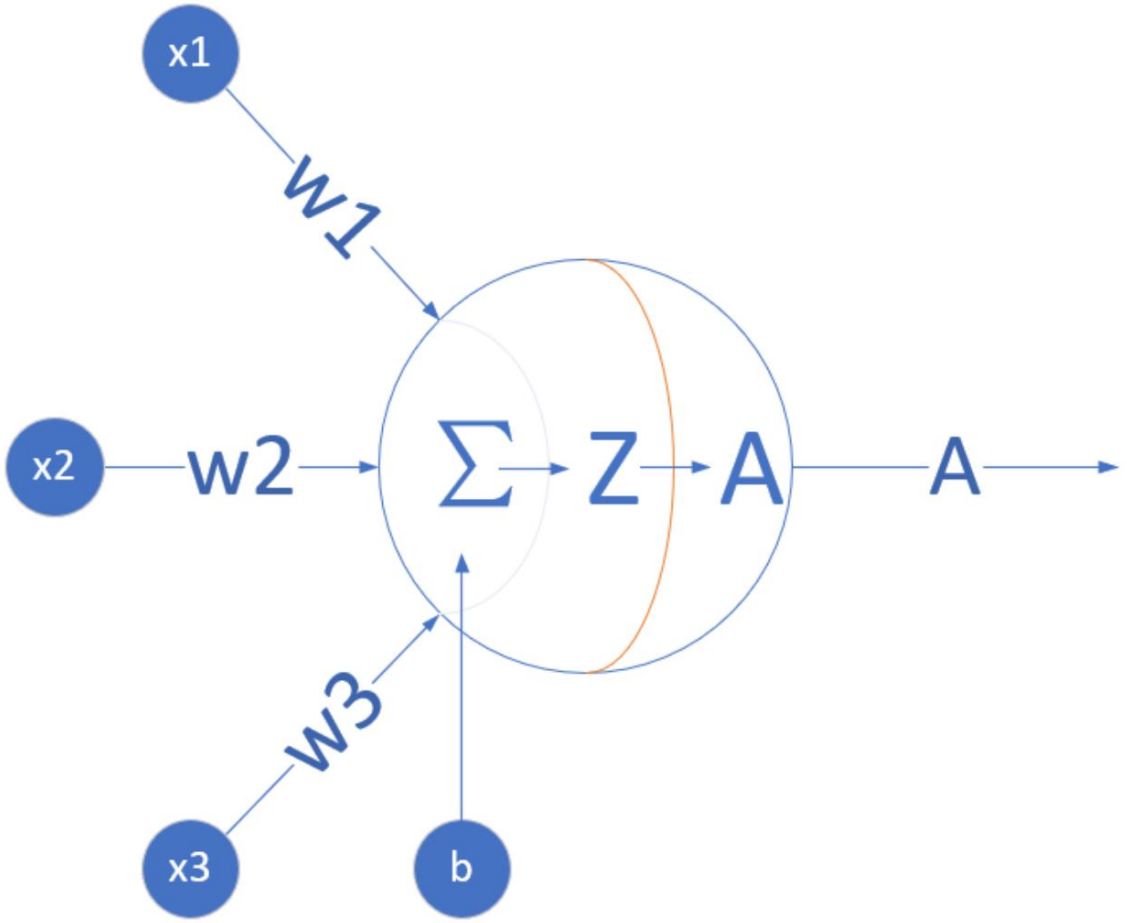
$$w_1x_1 + w_2x_2 + w_3x_3 - t \geq 0$$

$$w_1x_1 + w_2x_2 + w_3x_3 + b \geq 0$$

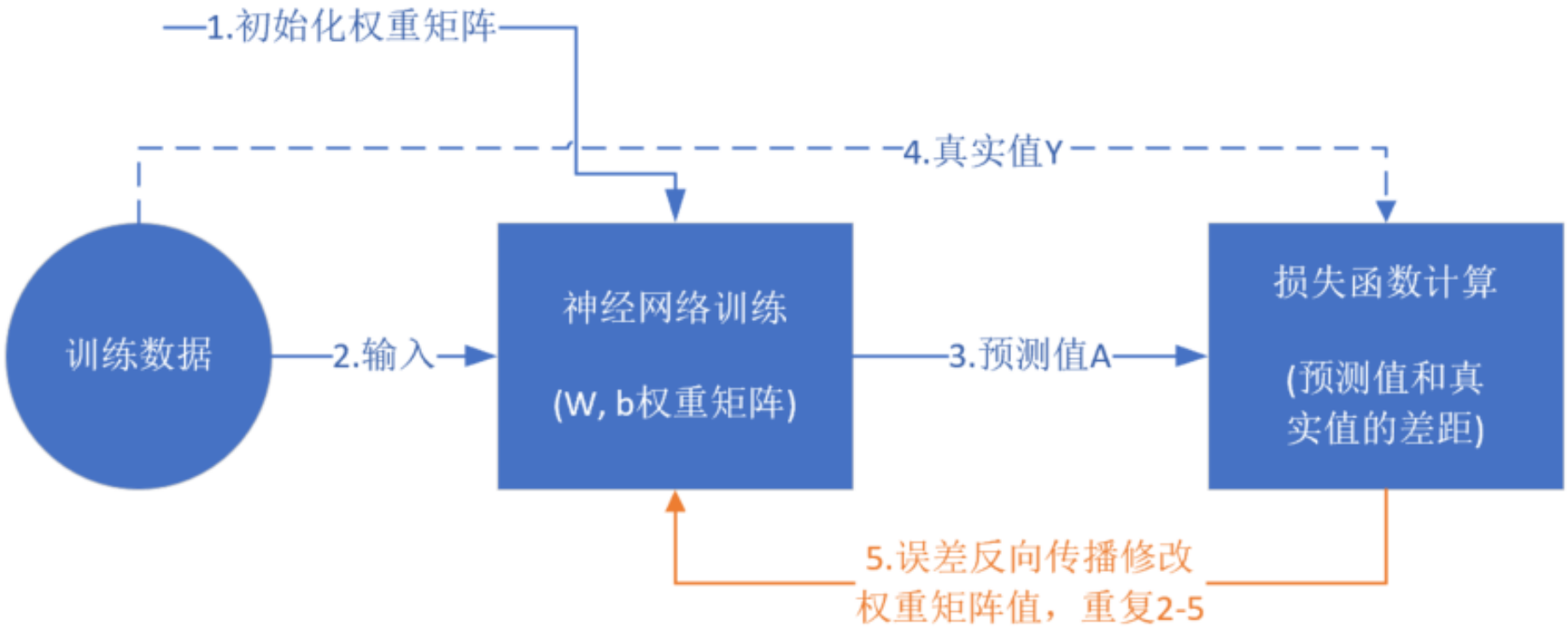
$$Z = w_1x_1 + w_2x_2 + w_3x_3 + b$$

神经网络的一些细节

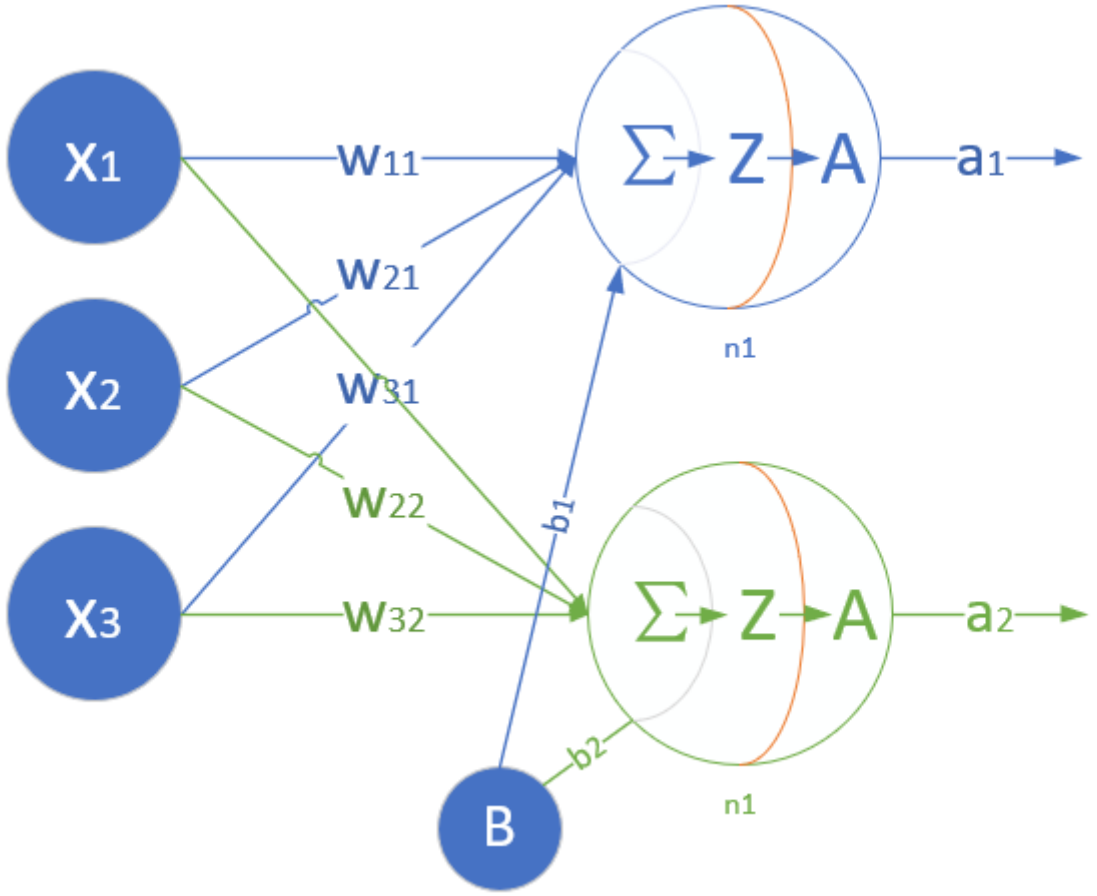
- 一个神经元可以有多个输入
- 一个神经元只能有一个输出，这个输出可以同时输入给多个神经元。
- 一个神经元的 w 的数量和输入的数量一致。
- 一个神经元只有一个 b。
- w 和 b 有人为的初始值，在训练过程中被不断修改。
- A 可以等于 Z，即激活函数不是必须有的。
- 一层神经网络中的所有神经元的激活函数必须一致。



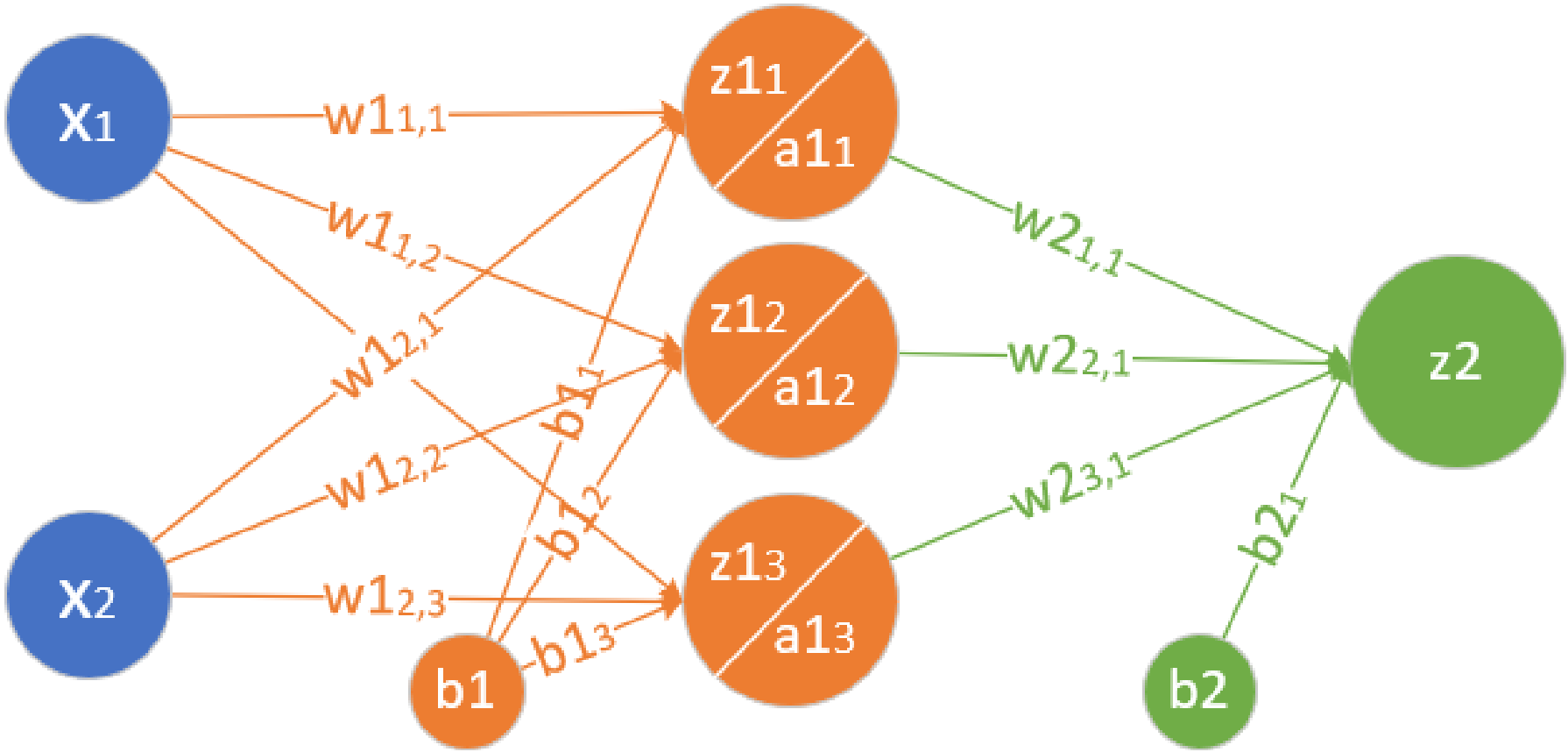
解剖神经网络训练过程


1.初始化权重矩阵
4.真实值Y
训练数据
2.输入
神经网络训练
(W, b权重矩阵)
3.预测值A
损失函数计算
(预测值和真实值的差距)
5.误差反向传播修改权重矩阵值，重复2-5

一个单层神经网络



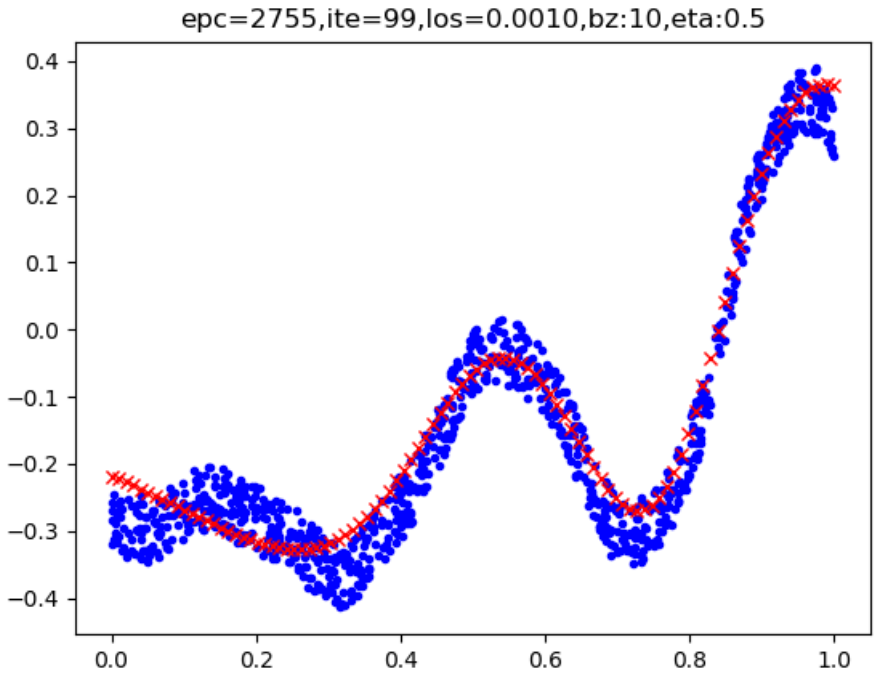
两层神经网络(矩阵运算)


x1
x2
b1
w11,1
w11,2
w12,1
w12,2
w12,3
b11
b12
b13
z11
a11
z12
a12
z13
a13
w21,1
w22,1
w23,1
b21
z2
b2

神经网络的主要功能

回归（Regression）或者叫做拟合（Fitting）

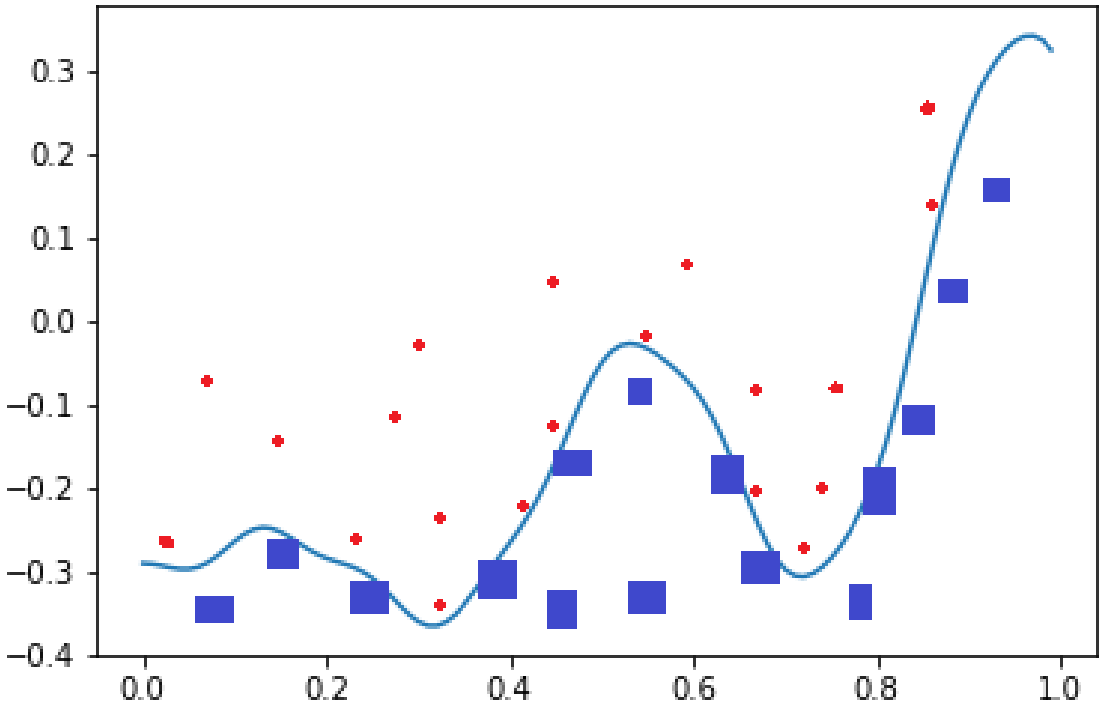
单层的神经网络能够模拟一条二维平面上的直线，从而可以完成线性分割任务。而理论证明，两层神经网络可以无限逼近任意连续函数



所谓回归或者拟合，其实就是给出x值输出y值的过程，并且让y值与样本数据形成的曲线的距离尽量小，可以理解为是对样本数据的一种骨架式的抽象。

神经网络的主要功能

分类（Classification）

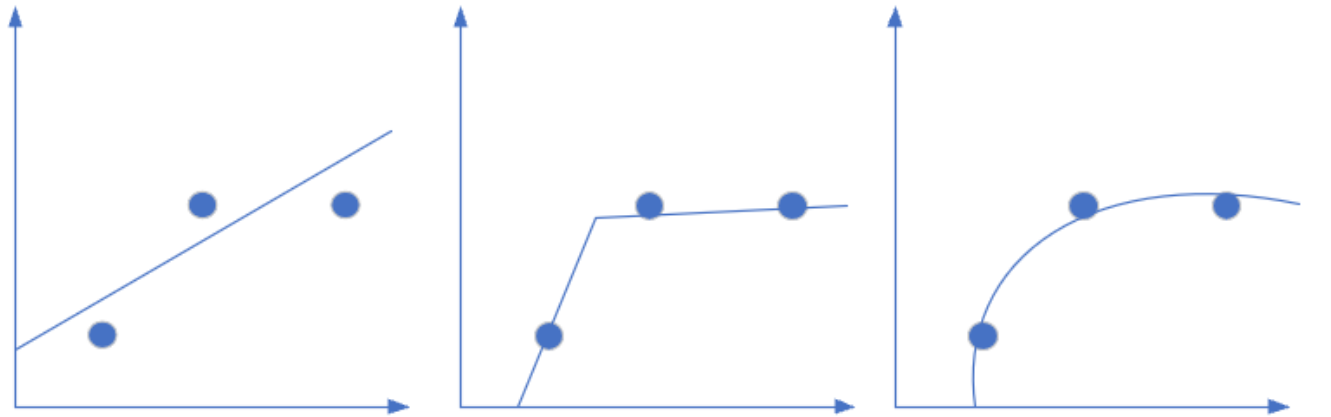


使用一个两层的神经网络可以得到一个非常近似的结果，使得分类误差在满意的范围之内。图中那条淡蓝色的曲线，本来并不存在，是通过神经网络训练出来的分界线，可以比较完美地把两类样本分开，所以分类可以理解为是对两类或多类样本数据的边界的抽象。

再来谈谈激活函数

我们不运用激活函数的话，则输出信号将仅仅是一个简单的线性函数。线性函数一个一级多项式。线性方程是很容易解决的，但是它们的复杂性有限，并且从数据中学习复杂函数映射的能力更小。一个没有激活函数的神经网络将只不过是一个线性回归模型罢了，不能解决现实世界中的大多数非线性问题。

没有激活函数，我们的神经网络将无法学习和模拟其他复杂类型的数据



线性拟合　　分段线性拟合　　有激活函数的拟合

最左侧的是线性拟合，中间的是分段线性拟合，右侧的是曲线拟合，只有当使用激活函数时，才能做到完美的曲线拟合。

三. 神经网络三大概念

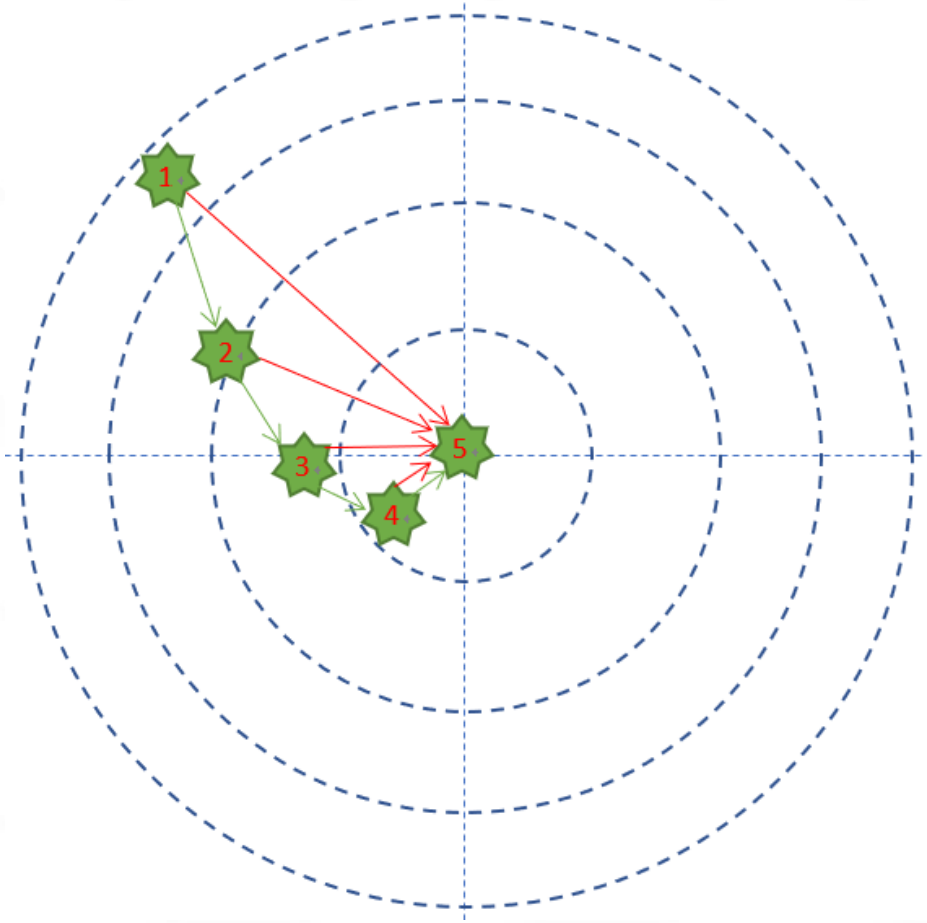
神经网络三大概念

神经网络训练的最基本的思想就是：先“**猜**”一个结果，称为预测结果 a，看看这个预测结果和事先标记好的训练集中的真实结果 y 之间的差距，然后调整策略，再试一次，这一次就不是“**猜**”了，而是有依据地向正确的方向靠近。如此反复多次，一直到预测结果和真实结果之间相差无几，亦即 $|a-y| \to 0$，就结束训练。

在神经网络训练中，我们把“**猜**”叫做初始化，可以随机，也可以根据以前的经验给定初始值。即使是“**猜**”，也是有技术含量的。

反向传播，梯度下降，损失函数。

看一个例子



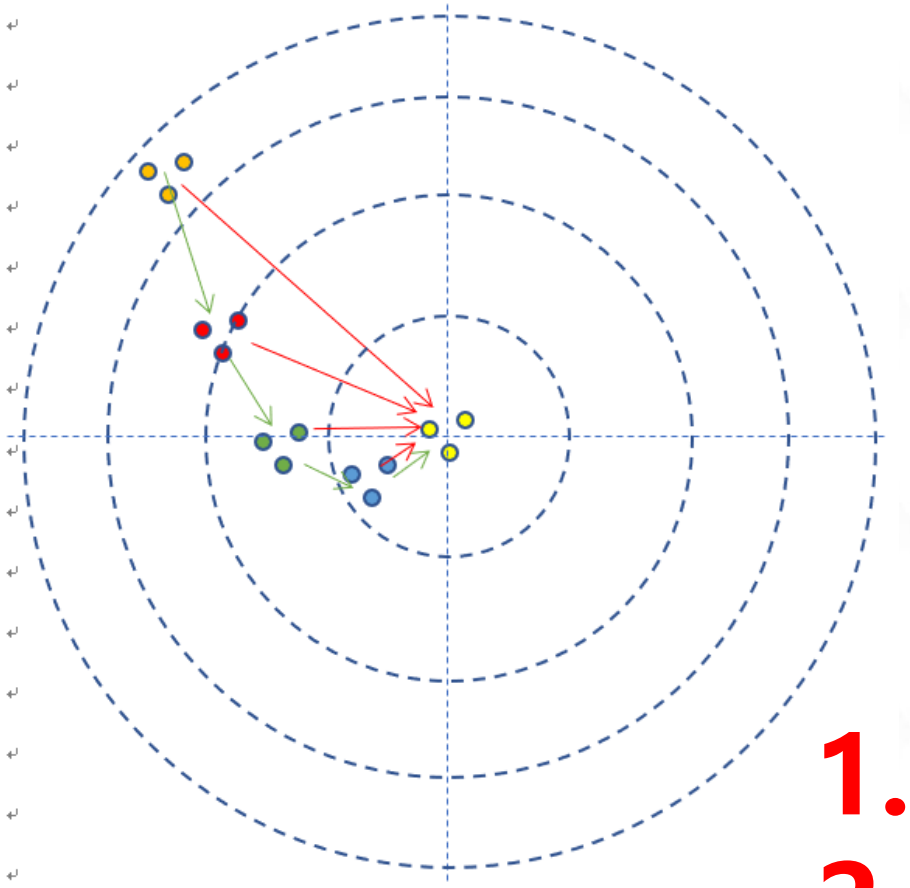
•每次试枪弹着点和靶心之间的差距就叫做误差，可以用一个误差函数来表示，比如差距的绝对值，如图中的红色线•。

•一共试枪5次，就是迭代/训练了5次的过程 。

•每次试枪后，把靶子拉回来看弹着点，然后调整下一次的射击角度的过程，叫做**反向传播**。注意，把靶子拉回来看和跑到靶子前面去看有本质的区别，后者容易有生命危险，因为还有别的射击者。一个不恰当的比喻是，在数学概念中，人跑到靶子前面去看，叫做正向微分；把靶子拉回来看，叫做反向微分。

•每次调整角度的数值和方向，叫做**梯度**。比如向右侧调整1毫米，或者向左下方调整2毫米。如图中的绿色矢量线。

看一个例子



如果每次3发子弹连发，这3发子弹的弹着点和靶心之间的差距之和再除以3，叫做**损失**，可以用损失函数来表示

梯度，是个矢量

1.距离；
2.方向

反向传播与梯度下降的工作原理

1.初始化；

2.正向计算；

3.损失函数为我们提供了计算损失的方法

4.梯度下降是在损失函数基础上向着损失最小的点靠近而指引了网络权重调整的方向；

5.反向传播把损失值反向传给神经网络的每一层，让每一层都根据损失值反向调整权重；

6.Go to 2，直到精度足够好（比如损失函数值小于 0.001）。

反向传播

梯度下降

梯度下降

梯度下降的数学公式：

$$\theta_{n+1} = \theta_n - \eta \cdot \nabla J(\theta)$$

其中：

- θ_{n+1}：下一个值；
- θ_n：当前值；
- $-$：减号，梯度的反向；
- η：学习率或步长，控制每一步走的距离，不要太快以免错过了最佳景点，不要太慢以免时间太长；
- ∇：梯度，函数当前位置的最快上升点；
- $J(\theta)$：函数。

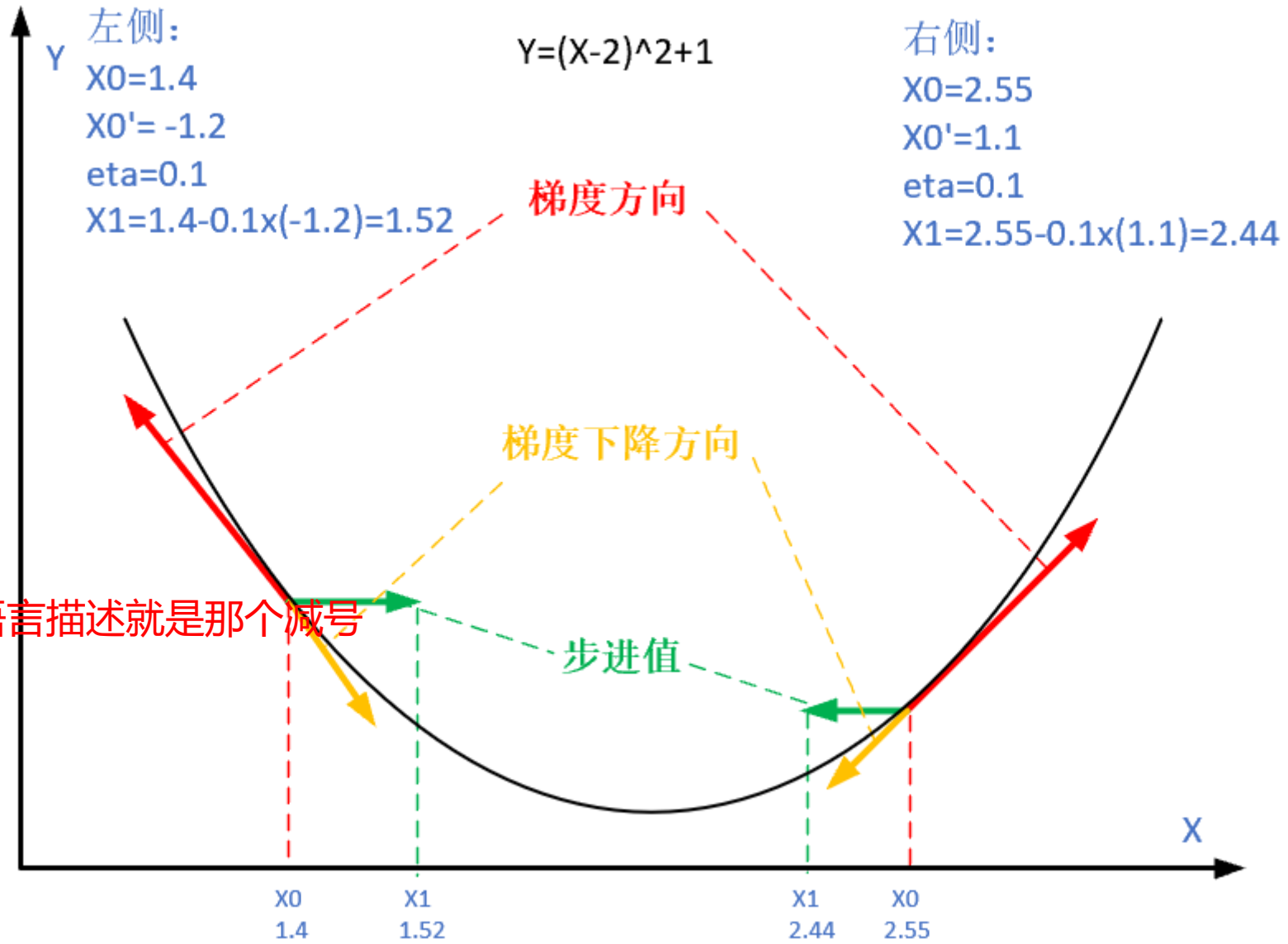
1.当前点；
2.方向；
3.步长

梯度下降

“梯度下降”包含了两层含义：

1.梯度：函数当前位置的最快上升点；

2.下降：与导数相反的方向，用数学语言描述就是那个减号



损失函数

损失函数的作用

损失函数的作用，就是计算神经网络每次迭代的前向计算结果与真实值的差距，从而指导下一步的训练向正确的方向进行。

如何使用损失函数呢？具体步骤：

1.用随机值初始化前向计算公式的参数；

2.代入样本，计算输出的预测值；

3.用损失函数计算预测值和标签值（真实值）的误差；

4.根据损失函数的导数，沿梯度最小方向将误差回传，修正前向计算公式中的各个权重值；

5.进入第2步重复，直到损失函数值达到一个满意的值就停止迭代。

损失函数

符号规则：a 是预测值，y 是样本标签值，$loss$ 是损失函数值。

- Gold Standard Loss，又称0-1误差 $$ loss=\begin{cases} 0 & a=y \\ 1 & a \ne y \end{cases} $$
- 绝对值损失函数

$$loss = |y - a|$$

- Hinge Loss，铰链/折页损失函数或最大边界损失函数，主要用于SVM（支持向量机）中

$$loss = \max(0, 1 - y \cdot a) \qquad y = \pm 1$$

- Log Loss，对数损失函数，又叫交叉熵损失函数(cross entropy error)

$$loss = -[y \cdot \ln(a) + (1 - y) \cdot \ln(1 - a)] \qquad y \in 0, 1$$

- Squared Loss，均方差损失函数 $$ loss=(a-y)^2 $$
- Exponential Loss，指数损失函数 $$ loss = e^{-(y \cdot a)} $$

损失函数 - 交叉熵

交叉熵（Cross Entropy）是Shannon信息论中一个重要概念，主要用于度量两个概率分布间的差异性信息。在信息论中，交叉熵是表示两个概率分布 p,q 的差异，其中 p 表示真实分布，q 表示预测分布，那么 H(p,q) 就称为交叉熵：

$$H(p,q)=\sum_i p_i \cdot \ln\frac{1}{q_i}=-\sum_i p_i \ln q_i$$

交叉熵可在神经网络中作为损失函数，p 表示真实标记的分布，q 则为训练后的模型的预测标记分布，交叉熵损失函数可以衡量 p 与 q 的相似性。

交叉熵函数常用于逻辑回归(logistic regression)，也就是分类(classification)。

损失函数 – 均方差

该函数就是最直观的一个损失函数了，计算预测值和真实值之间的欧式距离。预测值和真实值越接近，两者的均方差就越小。

均方差函数常用于线性回归(linear regression)，即函数拟合(function fitting)。公式如下：

$$loss = \frac{1}{2}(z-y)^2 \quad \text{(单样本)}$$

$$J = \frac{1}{2m}\sum_{i=1}^{m}(z_i - y_i)^2 \quad \text{(多样本)}$$

六. 小结

Microsoft

Reactor

Thank You!