一文搞懂损失函数（上）

P9工作法
分布式架构、上百人团队管理、全球支付实践与AI技术

在三步构建神经网络中已经提到损失函数的概念，但并没有说得特别透彻，这里对损失函数做一个详尽的分析，分为上下两篇。

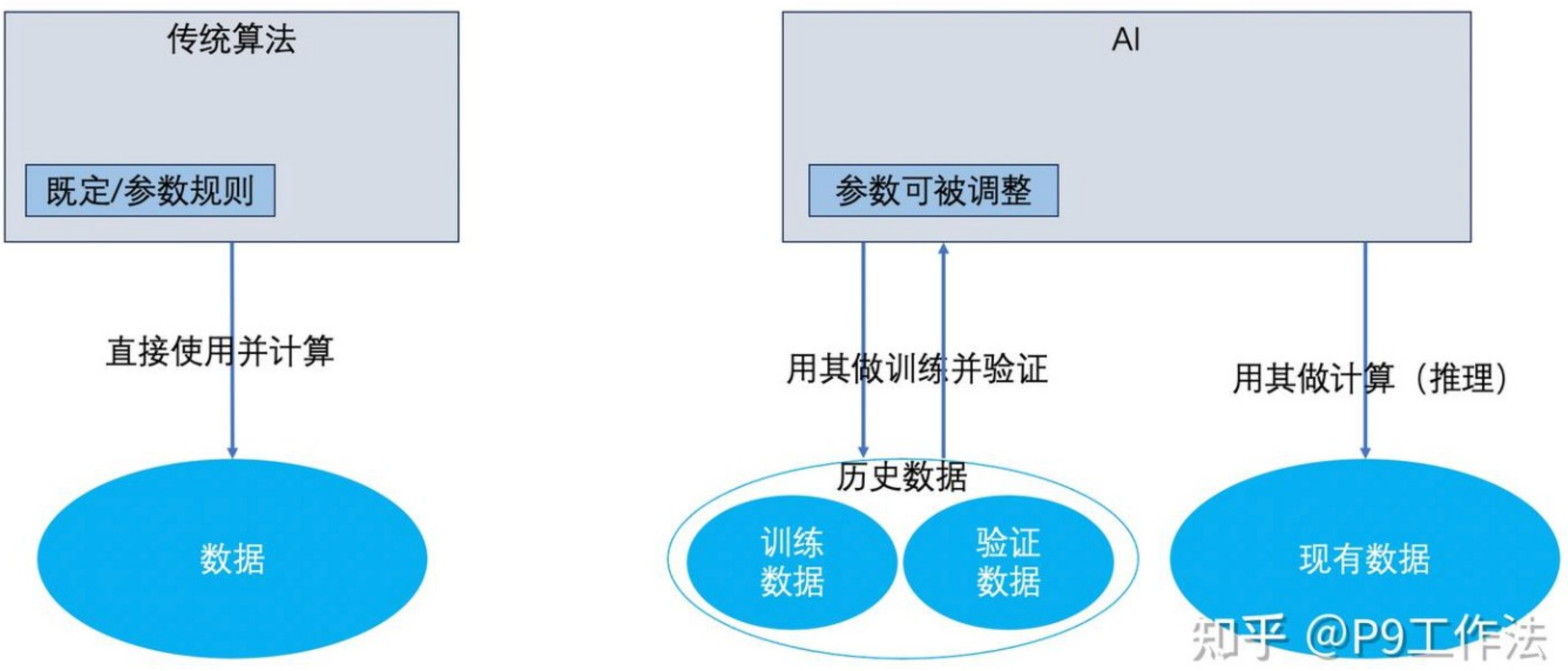
在分析损失函数之前，我们先回到原点来推导一下，为什么一定需要损失函数。

为什么需要损失函数

从AI与传统算法的区别谈起

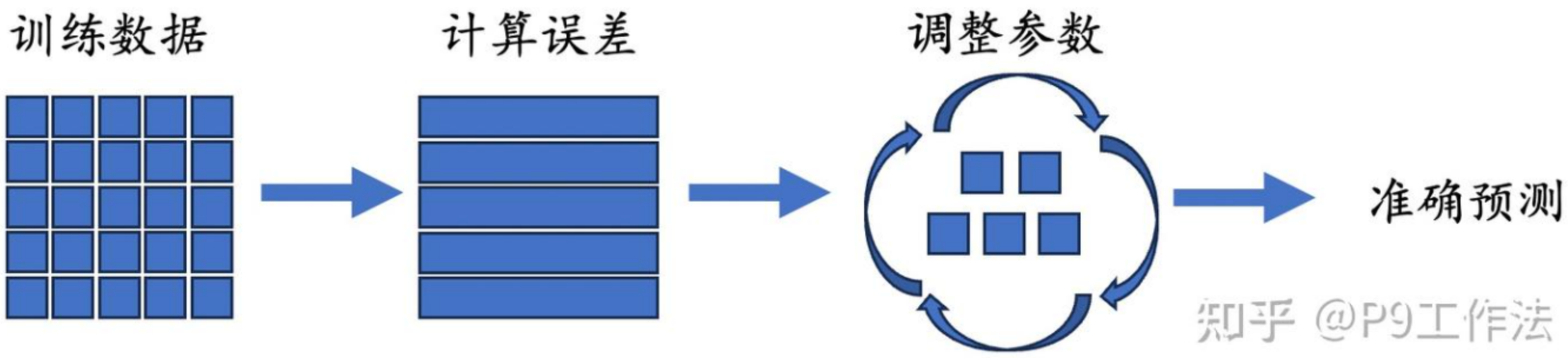
AI（特别是机器学习和深度学习）与传统算法之间的一个重要区别就是AI是把训练和推理（或计算）这两个过程分开的，而传统算法是基于既定规则对现有数据进行直接计算。传统算法的参数都是固定的，一般不需要调整。比如二叉树算法✦，数据来了就直接计算。

而AI模型是从大量历史数据中找规律，来学习其中的参数。会将数据分成两个小类，第一个是训练数据，顾名思义就是用该数据来训练我们的模型，使得我们的模型能够得到最优的参数集合。第二个是验证数据，用该数据来验证我们模型的学习效果。当模型得到了最优的参数集合后，就可以对现有数据进行计算（预测或者分类等）。



从模型的训练过程来看

既然模型是需要通过训练才能够得到，那么可以从训练的过程来做一个解读，一般分为以下几个步骤：



1、得到训练数据，这个工作看起来就几个字，但实际上非常有讲究，一个是如何得到大量的数据，一个是如何得到高质量的数据，当你上了几个T的数据肯定是不可能肉眼去看数据的质量。

2、假设对模型的参数进行初始化（如满足高斯分布✦的随机初始化），那么输入数据 x ，理论上应该可以得到一个预测结果数据 $\hat{y}$, 那么结果数据 $\hat{y}$ 的与真实结果值y(这里我们考虑是监督学习，训练数据是提供真实值的）之间的差值就是误差。

3、既然得到了误差，就可以通过误差去反向调整模型的参数，看调哪个参数会让误差小一点，当最终经过N次调整（调整过程可以看这个文章从结果推导原因-反向传播，一文搞懂梯度下降）后，就得到一个相对靠谱的模型参数。

4、当得到了靠谱的参数后，该模型就可用了，可以用验证数据进行验证，看该模型是不是靠谱。但实际上，不管怎么调整参数，都会有两种趋势性结果：

- 一个是模型对训练数据的拟合非常好，但是现有数据应用效果不好，道理也蛮简单，历史不代表未来，通过经验总结得到的未必适用于未来的新情况，这就叫做泛化性差。
- 一个是模型对训练数据效果一般，这就是欠拟合✦。这种情况也说不好对未来数据的拟合就一定行。

综上所得，其实损失函数就是预测值与真实值之间的差值，只是该差值用函数表达出来，这就是损失函数。损失函数是用来指导如何训练得到最佳参数的。可以说，找到了损失函数也就是找到了神经网络优化的目标。

什么是损失函数

在函数是神经网络的本质中提到，如果预测房价可以用函数 $y = Wx + b$ 来拟合这些数据。那么如何衡量这条直线好不好呢？显然可以想到以下方法：

- 预测值 $\hat{y}$ 与真实值 y 的差绝对值总和均值最小，这就是MAE（Mean Absolute Error）方法。
- 预测值 $\hat{y}$ 与真实值 y 的差平方总和均值最小，这就是MSE（Mean Squared Error）方法。

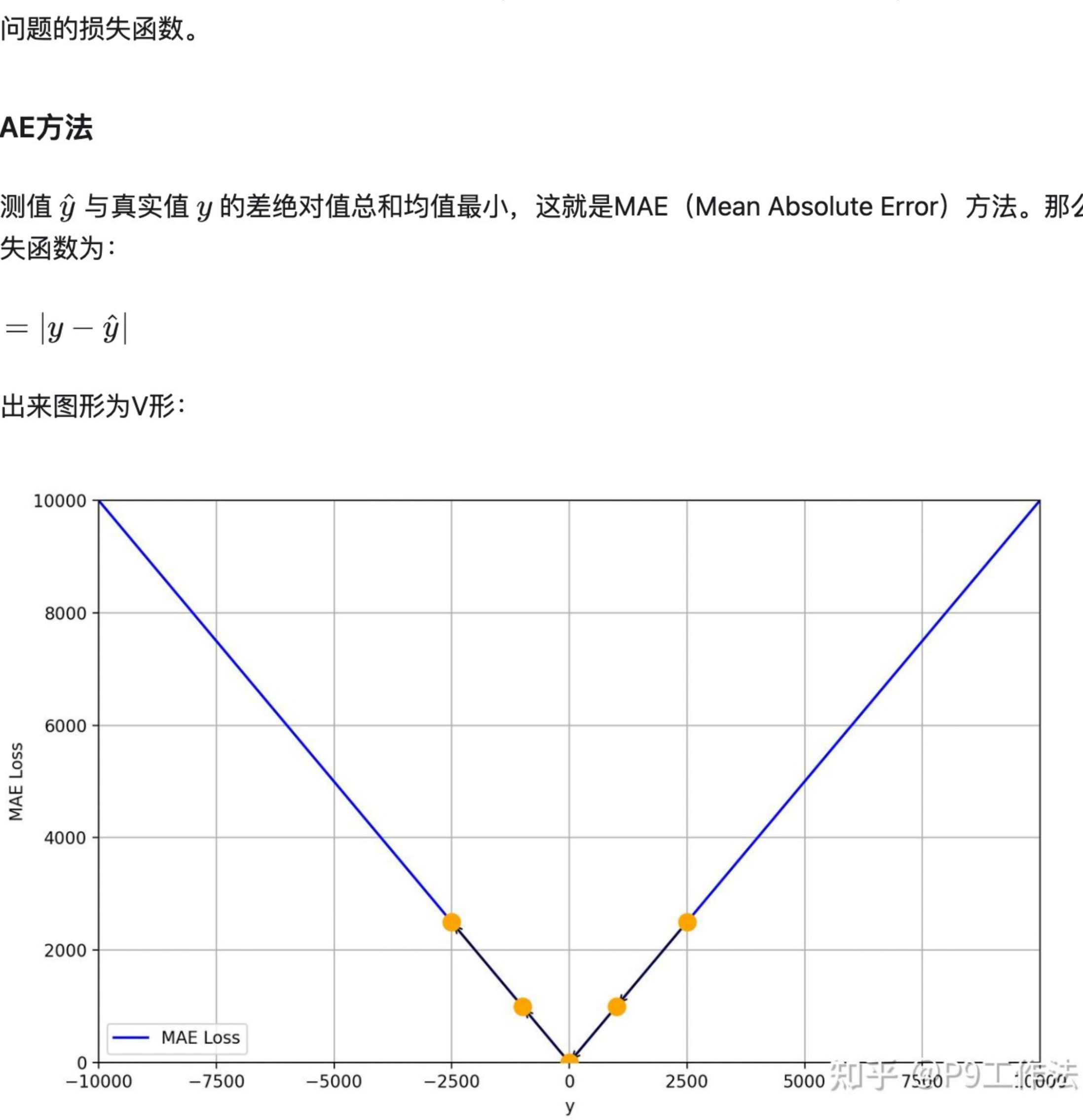
在 看全局知AI 文章中，把神经网络的任务可以分为回归，分类，聚类、生成四类。但归根结底可以认为是回归问题和分类问题。为了便于阅读，这里只介绍回归问题的损失函数，下一篇再介绍分类问题的损失函数。

MAE方法

预测值 $\hat{y}$ 与真实值 y 的差绝对值总和均值最小，这就是MAE（Mean Absolute Error）方法。那么损失函数为：

$$L = |y - \hat{y}|$$

画出来图形为V形：



当 $\hat{y} < y$ 时，$L(\hat{y}) = y - \hat{y}$，这是一个斜率为 -1 的直线。

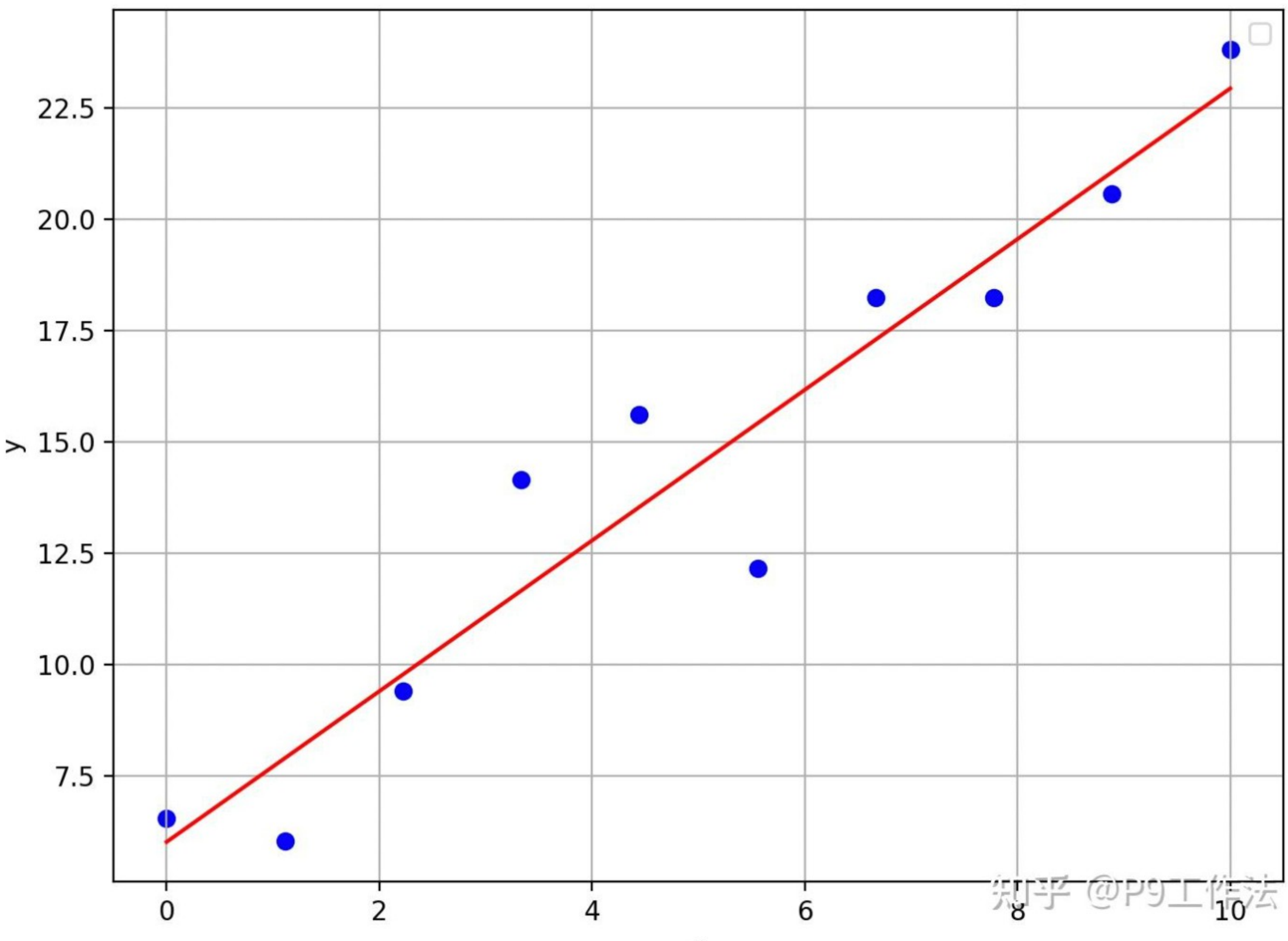
当 $\hat{y} > y$ 时，$L(\hat{y}) = \hat{y} - y$，这是一个斜率为 +1 的直线。

当 $\hat{y} = y$ 时， $L(\hat{y}) = 0$。

这样的损失函数有较大的缺陷，一个是0点不可导，二是一个梯度固定，一个很小的预测误差（损失）就会有较大的梯度，不好收敛。

MSE方法

MSE也叫最小二乘法✦。最小二乘法（Least Squares Method）是一种用于拟合数据的统计方法，其目标是找到一条最佳拟合线或曲线，使得所有数据点到该线或曲线的距离的平方和最小。本质上与MSE是一样的，只是MSE除以了N，变成了均值。最小二乘这个翻译有点不贴切，其实叫最小平方法是更贴切的。



损失函数 $L = \sum(\hat{y}_i - y_i)^2 = \sum(wx_i + b - y_i)^2$

MSE-传统计算法

求当 L 最小时， W 与 b 的取值。 因为 x_i 与 y_i 都是训练数据，即为已知数。如：

x	1	2	3	4
y	0.8	1.5	1.8	2.0

那么把对应的 x_i 与 y_i 代入，就会得到一个关于 W, b 的二次函数。

则可以对W与b求偏导，使其等于0，求解出W与b，这样就会得到在 L 最小时的 W, b，即更好的拟合直线。

$\frac{\partial \mathrm{L}}{\partial W}$ = 60W + 20b - 34.4 = 0

$\frac{\partial \mathrm{L}}{\partial b}$ = 20W + 8b - 12.2 = 0

解得 W = 0.39, b = 0.55 即在 L 最小时，拟合直线为 $y = 0.39x + 0.55$

这里需要注意的是， x_i 与 y_i 都是训练数据，也就是知数，只是通过高等数学的求偏导就能够得到W与b的值。

MSE-矩阵计算法

上面的传统计算方法的确也是可以，但其实有更好的求解方法，那就是矩阵。

令 $X = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \end{bmatrix}$, $B = \begin{bmatrix} b \\ w \end{bmatrix}$,

则 $X \cdot B$，就是$\hat{y} = \begin{bmatrix} b + wx_1 \\ b + wx_2 \\ b + wx_3 \\ b + wx_4 \end{bmatrix}$，那么 $\hat{y} - y = e = \begin{bmatrix} y_1 - (b + wx_1) \\ y_2 - (b + wx_2) \\ y_3 - (b + wx_3) \\ y_4 - (b + wx_4) \end{bmatrix}$，则

$$L = e^T \cdot e = [\, y_1 - (b + wx_1), \quad y_2 - (b + wx_2), \quad \ldots\,] \cdot \begin{bmatrix} y_1 - (b + wx_1) \\ y_2 - (b + wx_2) \\ \vdots \\ y_4 - (b + wx_4) \end{bmatrix}$$

同样对矩阵求导✦，可以得到

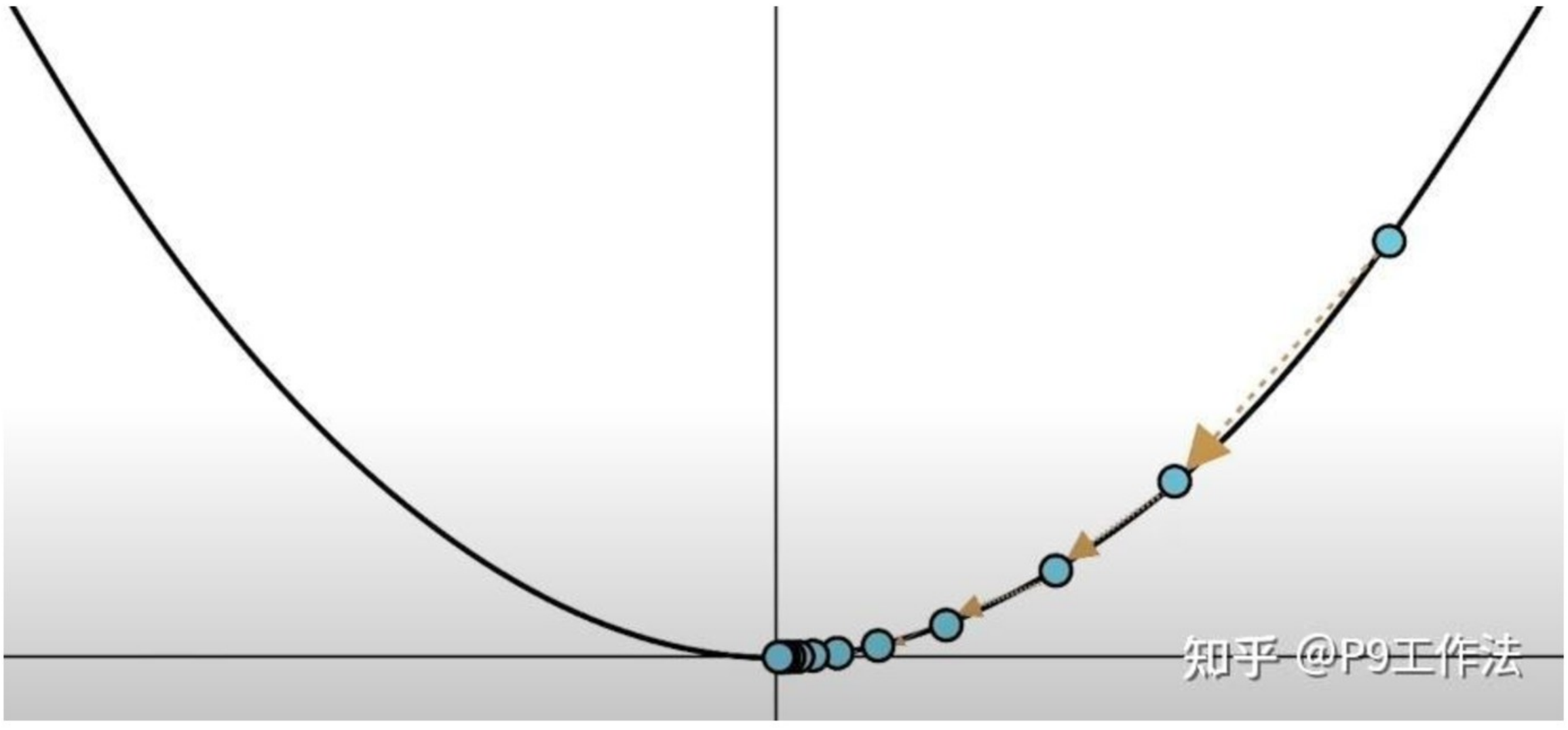
$$X^T \cdot X \cdot B = X^T \cdot y \Rightarrow B = (X^T \cdot X)^{-1} \cdot X^T \cdot y$$

同样可以解出来 W 和 b 的值。

从这里也可以看出来，为什么需要GPU，AI里面都是矩阵运算✦，而矩阵运算是最容易并行的，因为都是每一行与每一列对应数相乘再相加。

MAE与MSE的比较

MSE的损失函数图形其实是一个抛物线，如下图，在0处可导。



MAE与MSE比较下来，有以下区别：

1、MSE其实会放大大的误差，缩小小的误差，所以显然它最终的结果是会让函数去靠近偏差大的数据，也就是会去照顾到异常数据。

2、但是MAE不会这样，会更好去拟合正常点，反过来说这也可以用来去惩罚哪些异常的数据。

折中的方法

既然MAE和MSE都有各自的优缺点，是不是有折中的损失函数。的确是有的，那就是Huber Loss。Huber 损失函数的设计目的是为了提高对异常值的鲁棒性✦，同时保持损失函数的可微性。函数表达为：

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta \\ \delta(|a| - \frac{1}{2}\delta) & \text{for } |a| > \delta \end{cases}$$

，其中 $a = y - \hat{y}$ ，而 δ是一个超参数✦。

当预测误差小于阈值（超参数）的时候就是MSE（上面的式子），如果预测误差大于阈值（超参数）的类似于MAE（下面的式子），这个函数在0也是可导的。

如下图所示：