Microsoft

Reactor

一起学人工智能系列
- 线性回归1

2021-09-01

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

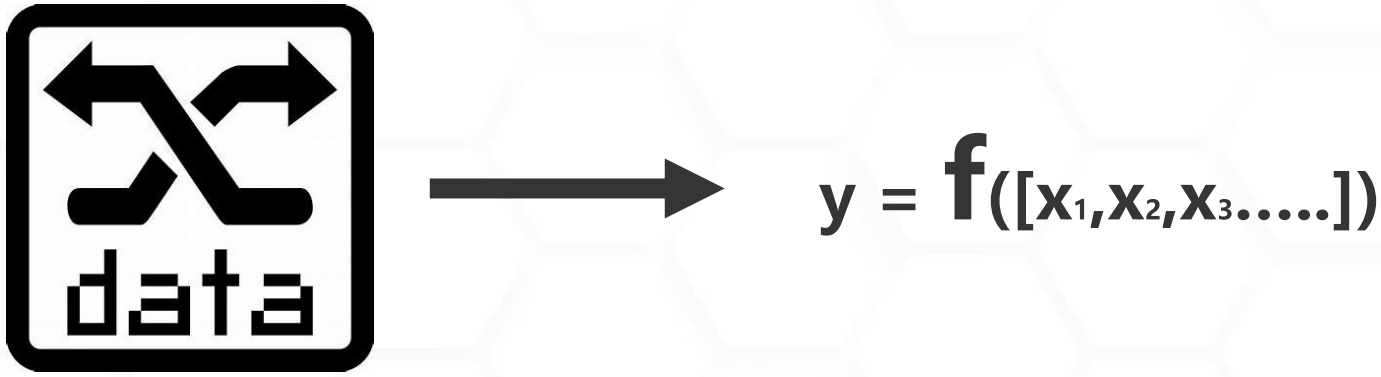
Email : kinfeylo@microsoft.com **Blog :** https://blog.csdn.net/kinfey

Twitter : @Ljh8304

一．线性回归

一切从监督学习谈起

Supervised Learning – 预测未来



经验决定一切，整合大量的标注数据，应用在预测价格，判断分类等场景上

需要找到一个基于特征数据，生成结果的方法

一般都有多个特征数据

x值拟合到计算中，从而为训练数据集中的所有情况合理准确地生成y。

回归 – 起源

“回归”是由英国著名生物学家兼统计学家高尔顿(Francis Galton,1822～1911.生物学家达尔文的表弟)在研究人类遗传问题时提出来的。

1855年， 高尔顿发表《遗传的身高向平均数方向的回归》一文，他和他的学生卡尔•皮尔逊Karl·Pearson通过观察1078对夫妇的身高数据，以每对夫妇的平均身高作为自变量，取他们的一个成年儿子的身高作为因变量，分析儿子身高与父母身高之间的关系，发现父母的身高可以预测子女的身高，两者近乎一条直线。当父母越高或越矮时，子女的身高会比一般儿童高或矮，他将儿子与父母身高的这种现象拟合出一种线形关系，分析出儿子的身高y与父亲的身高x大致可归结为一下关系：

y=33.73+0.516*x (单位为英寸)

根据换算公式1英寸=0.0254米， 1米=39.37英寸。单位换算成米后：

Y= 0.8567+0.516*X （单位为米);

假如父母辈的平均身高为1.75米，则预测子女的身高为1.7597米。

这种趋势及回归方程表明父母身高每增加一个单位时，其成年儿子的身高平均增加0.516个单位。这就是回归一词最初在遗传学上的含义。

回归 – 以数学观点出发

回归，指研究一组随机变量(Y1 ，Y2 ，…，Yi)和另一组(X1，X2，…，Xk)变量之间关系的统计分析方法，又称多重回归分析。通常Y1，Y2，…，Yi是因变量，X1、X2，…，Xk是自变量。

回归分析是一种数学模型。

①从一组数据出发，确定某些变量之间的定量关系式；即建立数学模型并估计未知参数。通常用最小二乘法。

②检验这些关系式的可信任程度。

③在多个自变量影响一个因变量的关系中，判断自变量的影响是否显著，并将影响显著的选入模型中，剔除不显著的变量。通常用逐步回归、向前回归和向后回归等方法。

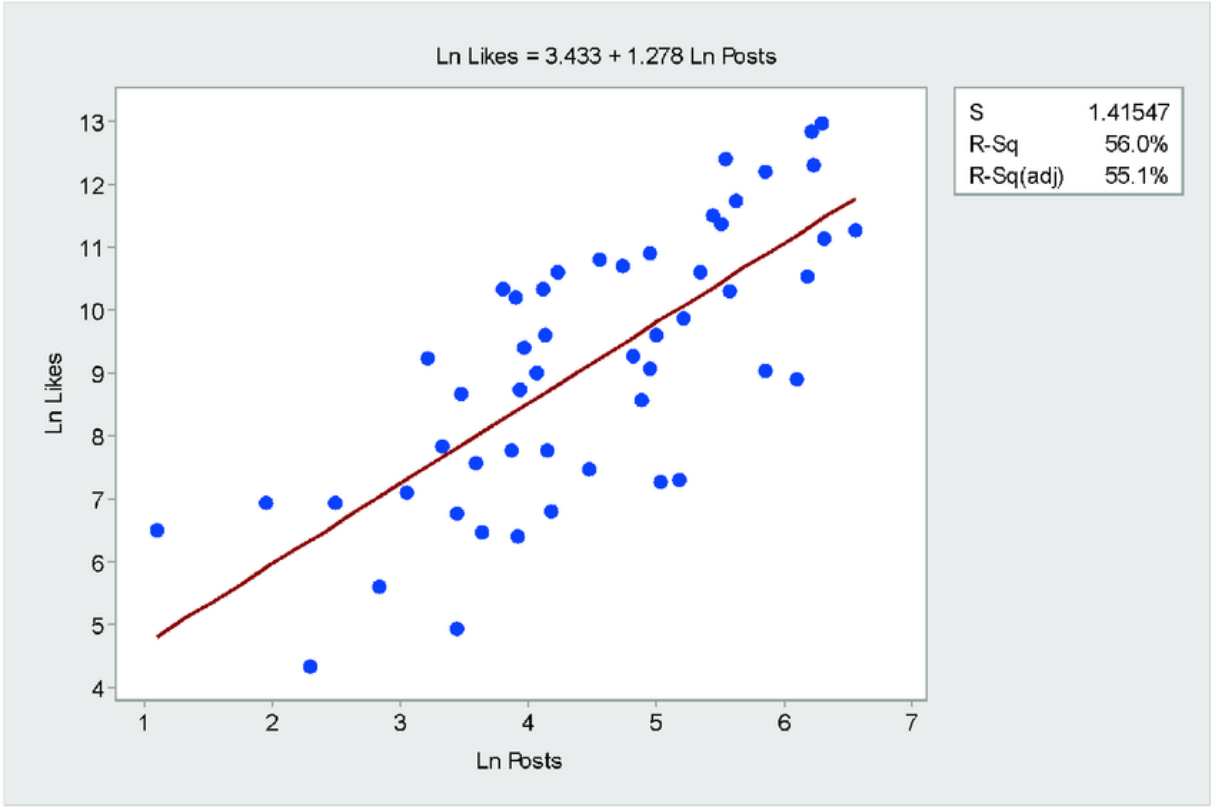
④利用所求的关系式对某一过程进行预测或控制。

回归

回归主要的种类有：线性回归、曲线回归、二元logistic回归、多元logistic回归。

常用的一些应用场景

1. 销售量预测
2. 制造缺陷预测
3. 预测名人的离婚率
4. 预测所在地区的房价



线性回归

回归,一般都是指线性回归(Linear Regression).

用线性回归作为学习神经网络的起点，是一个非常好的选择，因为线性回归问题本身比较容易理解，在它的基础上，逐步的增加一些新的知识点，会形成一条比较平缓的学习曲线，或者说是迈向神经网络的第一个小台阶。

单层的神经网络，其实就是一个神经元，可以完成一些线性的工作，比如拟合一条直线，这用一个神经元就可以实现。当这个神经元只接收一个输入时，就是单变量线性回归，可以在二维平面上用可视化方法理解。当接收多个变量输入时，叫做多变量线性回归，此时可视化方法理解就比较困难了，通常我们会用变量两两组对的方式来表现。

当变量多于一个时，两个变量的量纲和数值有可能差别很大，这种情况下，我们通常需要对样本特征数据做归一化，然后把数据喂给神经网络进行训练，否则会出现“消化不良”的情况。

线性回归示例

如图：y = ax+b, 带入两点，
求得解析解为y = 2x + 1

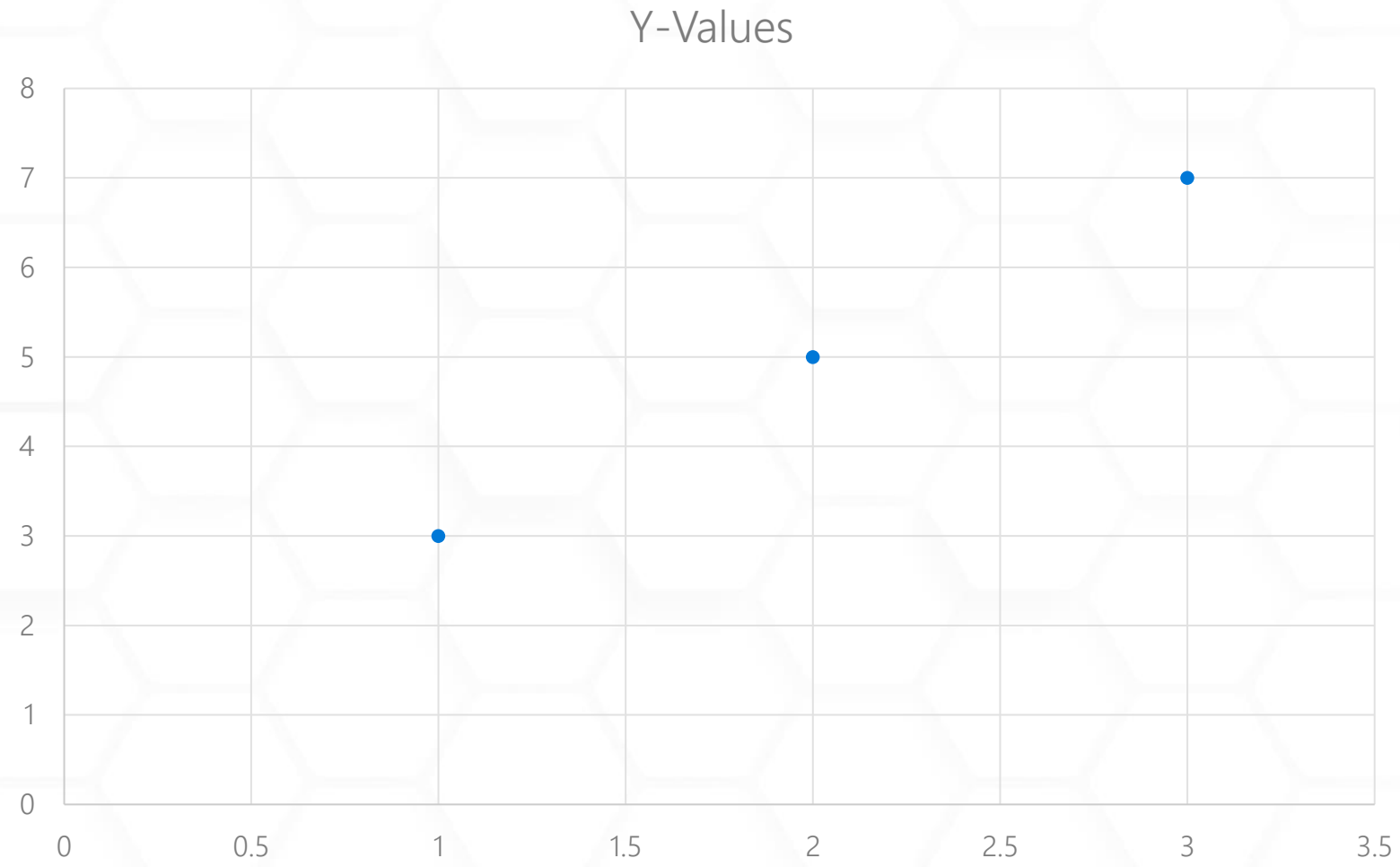
回归方程 (regression equation)

y = y = 2x + 1

回归系数 (regression weights)

2和1可以称为回归系数

回归: 求这些回归系数的过程

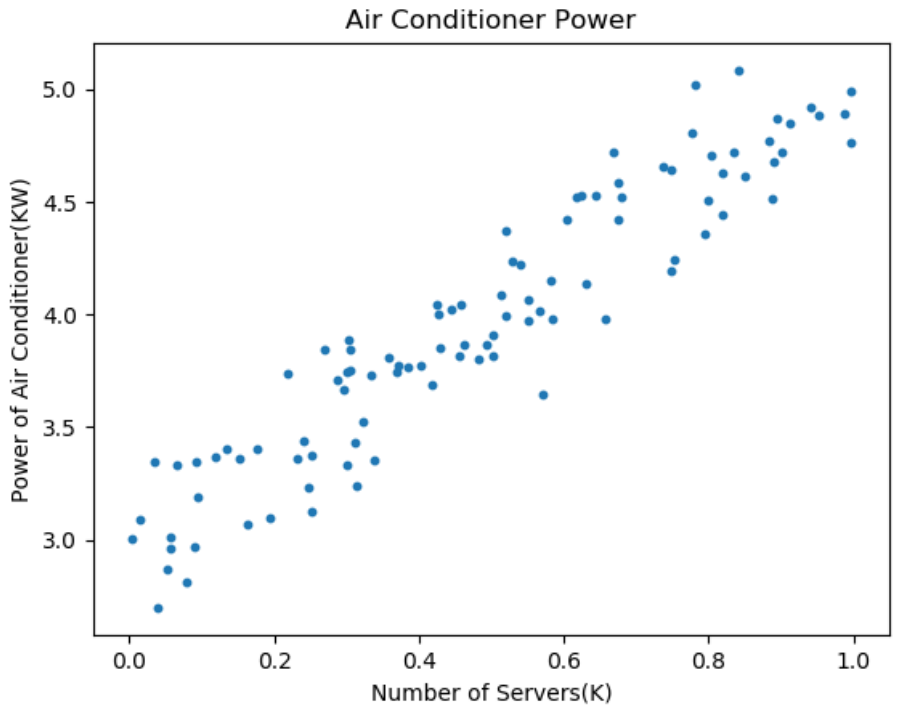


二. 单变量线性回归

场景

在互联网建设初期，各大运营商需要解决的问题就是保证服务器所在的机房的温度常年保持在23摄氏度左右。在一个新建的机房里，如果计划部署346台服务器，我们如何配置空调的最大功率？

样本序号	服务器数量(千台)X	空调功率(千瓦)Y
1	0.928	4.824
2	0.469	2.950
3	0.855	4.643
...	...	...



单变量线性回归问题

回归分析是一种数学模型。当因变量和自变量为线性关系时，它是一种特殊的线性模型。

最简单的情形是一元线性回归，由大体上有线性关系的一个自变量和一个因变量组成，模型是：

$$y = \mathrm{a} + \mathrm{b}X + \varepsilon$$

X是自变量，Y 是因变量，ε 是随机误差，a 和 b 是参数，在线性回归模型中，a,b 是要通过算法学习出来的

线性回归模型

对于线性回归模型，有如下一些概念需要了解：

•通常假定随机误差 ε 的均值为 0，方差为σ2（σ2>0，σ2 与 X 的值无关）

•若进一步假定随机误差遵从正态分布，就叫做正态线性模型

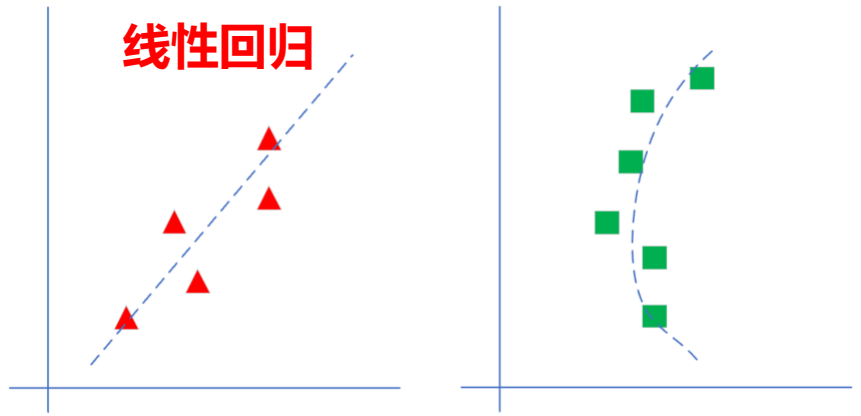
•一般地，若有 k 个自变量和 1 个因变量（即公式1中的 Y），则因变量的值分为两部分：一部分由自变量影响，即表示为它的函数，函数形式已知且含有未知参数；另一部分由其他的未考虑因素和随机性影响，即随机误差

•当函数为参数未知的线性函数时，称为线性回归分析模型

•当函数为参数未知的非线性函数时，称为非线性回归分析模型

•当自变量个数大于 1 时称为多元回归

•当因变量个数大于 1 时称为多重回归



三. 方法

解决方法

1. 最小二乘法;

2. 梯度下降法;

3. 简单的神经网络法;

4. 更通用的神经网络算法。

最小二乘法

小二乘法，也叫做最小平方法（Least Square），它通过最小化误差的平方和寻找数据的最佳函数匹配。利用最小二乘法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小。最小二乘法还可用于曲线拟合。其他一些优化问题也可通过最小化能量或最小二乘法来表达。

线性回归试图学得：

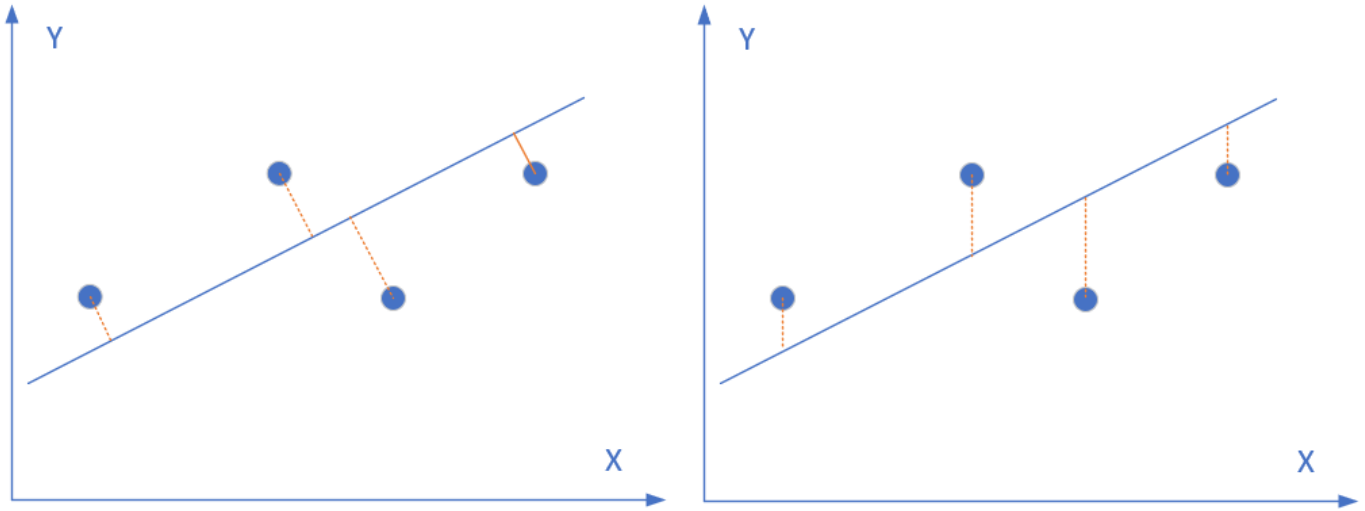
zi=w·xi+b -> zi≃yi

其中，xi 是样本特征值，yi 是样本标签值，zi 是模型预测值。

如何学得 w 和 b 呢？均方差(MSE - mean squared error)是回归任务中常用的手段：

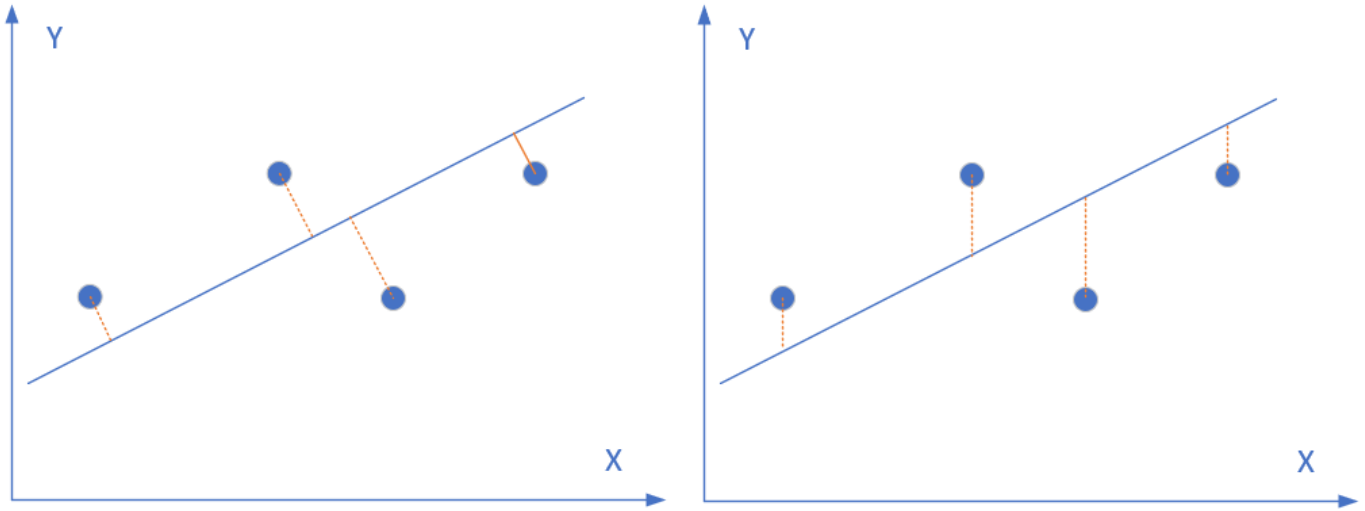
$$J = \sum_{i=1}^{m} (z(xi) - yi)^2 = \sum_{i=1}^{m} (yi - wxi - b)^2$$

最小二乘法



圆形点是样本点，直线是当前的拟合结果。如左图所示，我们是要计算样本点到直线的垂直距离，需要再根据直线的斜率来求垂足然后再计算距离，这样计算起来很慢；但实际上，在工程上我们通常使用的是右图的方式，即样本点到直线的竖直距离，因为这样计算很方便，用一个减法就可以了。

最小二乘法(MSE, Mean Square Error)



我们选择的参数决定了我们得到的直线相对于我们的训练集的准确程度，模型所预测的值与训练集中实际值之间的差距（红线所指）就是建模误差（modeling error）。

圆形点是样本点，直线是当前的拟合结果。如左图所示，我们是要计算样本点到直线的垂直距离，需要再根据直线的斜率来求垂足然后再计算距离，这样计算起来很慢；但实际上，在工程上我们通常使用的是右图的方式，即样本点到直线的竖直距离，因为这样计算很方便，用一个减法就可以了。

•拟合过程中因为w和b的取值准确度， 预测的结果与训练集中的实际值有差距。

梯度下降法

在下面的公式中，我们规定 x 是样本特征值（单特征），y 是样本标签值，z 是预测值，下标 i 表示其中一个样本。

预设函数（Hypothesis Function）

线性函数：

zi=w * xi+b

损失函数（Loss Function）

均方误差： $loss_i(w,b) = \frac{1}{2}(Zi - y_i)^2$

计算 w 的梯度

$$\frac{\partial loss}{\partial w} = \frac{\partial loss}{\partial z_i}\frac{\partial z_i}{\partial w} = (z_i - y_i)x_i$$

计算 b 的梯度

$$\frac{\partial loss}{\partial b} = \frac{\partial loss}{\partial z_i}\frac{\partial z_i}{\partial b} = z_i - y_i$$

与最小二乘法比较可以看到，梯度下降法和最小二乘法的模型及损失函数是相同的，都是一个线性模型加均方差损失函数，模型用于拟合，损失函数用于评估效果。

神经网络法

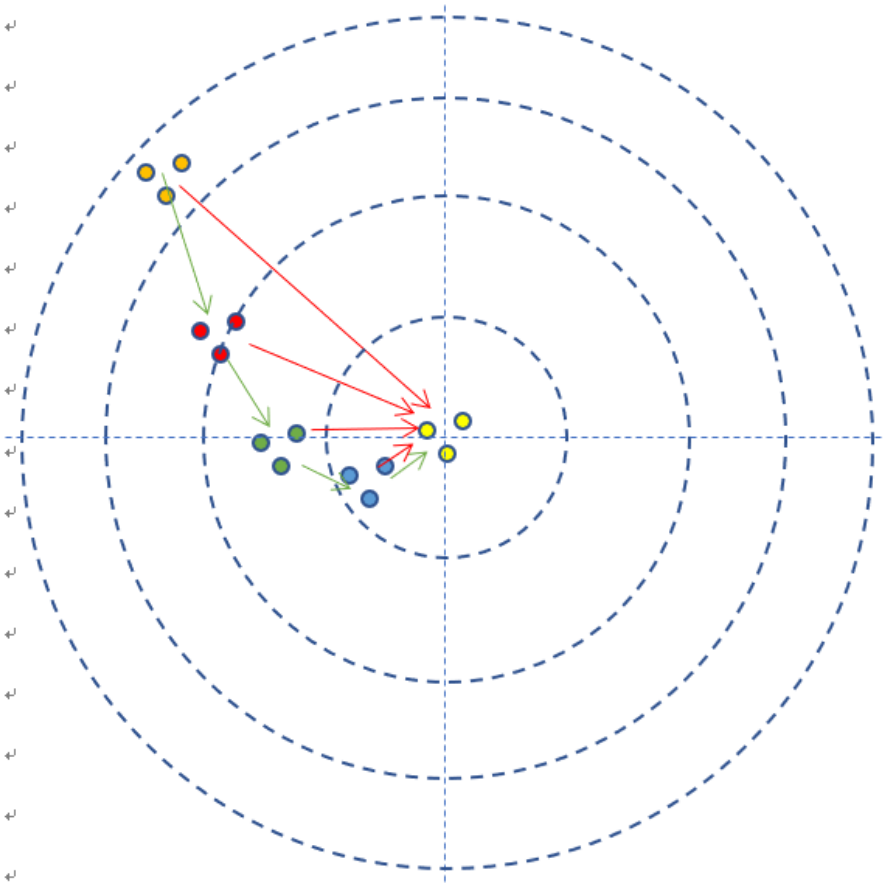
1.初始化权重值

2.根据权重值放出一个解

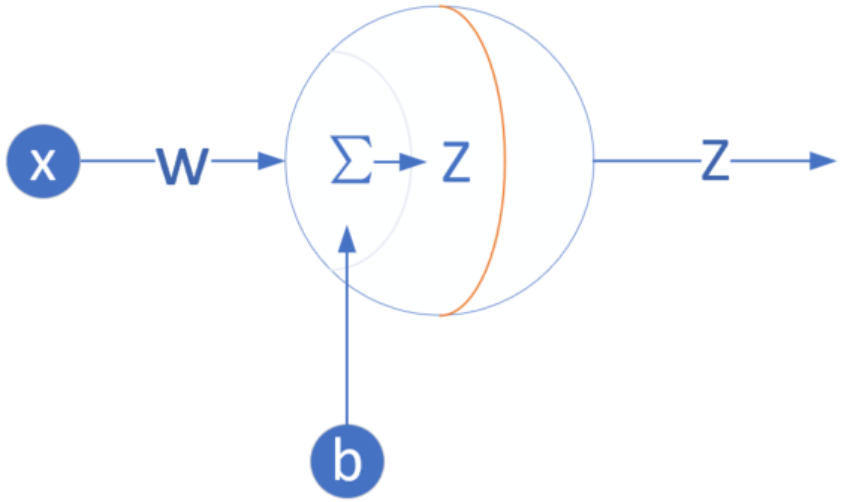
3.根据均方差函数求误差

4.误差反向传播给线性计算部分以调整权重值

5.是否满足终止条件？不满足的话跳回2



神经网络法



计算 w 的梯度

$$\frac{\partial loss}{\partial w} = \frac{\partial loss}{\partial z_i}\frac{\partial z_i}{\partial w} = (z_i - y_i)x_i$$

计算 b 的梯度

$$\frac{\partial loss}{\partial b} = \frac{\partial loss}{\partial z_i}\frac{\partial z_i}{\partial b} = z_i - y_i$$

输入层

此神经元在输入层只接受一个输入特征，经过参数 w,b 的计算后，直接输出结果。这样一个简单的“网络”，只能解决简单的一元线性回归问题，而且由于是线性的，我们不需要定义激活函数，这就大大简化了程序，而且便于大家循序渐进地理解各种知识点。严格来说输入层在神经网络中并不能称为一个层

权重 w,b

因为是一元线性问题，所以 w,b 都是标量。

输出层

输出层 1 个神经元，线性预测公式是：
zi=w*xi+b

z 是模型的预测输出，y 是实际的样本标签值，下标 i 为样本。

损失函数

因为是线性回归问题，所以损失函数使用均方差函数。

$$loss_i(w,b) = \frac{1}{2}(Zi - y_i)^2$$

四. 多样本计算

问题

单样本计算有一些缺点：

1. 前后两个相邻的样本很有可能会对反向传播产生相反的作用而互相抵消。假设样本1造成了误差为 0.5，w 的梯度计算结果是 0.1；紧接着样本2造成的误差为 −0.5，w 的梯度计算结果是 −0.1，那么前后两次更新 w 就会产生互相抵消的作用。

2. 在样本数据量大时，逐个计算会花费很长的时间。由于我们在本例中样本量不大（200个样本），所以计算速度很快，觉察不到这一点。在实际的工程实践中，动辄10万甚至100万的数据量，轮询一次要花费很长的时间。

前向计算

由于有多个样本同时计算，所以我们使用 x_i 表示第 i 个样本，X 是样本组成的矩阵，Z 是计算结果矩阵，w 和 b 都是标量：

$$Z = X \cdot w + b \tag{1}$$

把它展开成3个样本（3行，每行代表一个样本）的形式：

$$X = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

$$Z = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \cdot w + b = \begin{pmatrix} x_1 \cdot w + b \\ x_2 \cdot w + b \\ x_3 \cdot w + b \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix} \tag{2}$$

损失函数

用传统的均方差函数，其中，z 是每一次迭代的预测输出，y 是样本标签数据。我们使用 m 个样本参与计算，因此损失函数为：

$$J(w,b)=\frac{1}{2m}\sum_{i=1}^{m}(z_i-y_i)^2$$

其中的分母中有个2，实际上是想在求导数时把这个2约掉，没有什么原则上的区别。

我们假设每次有3个样本参与计算，即 $m=3$，则损失函数实例化后的情形是：

$$\begin{aligned}J(w,b)&=\frac{1}{2\times 3}[(z_1-y_1)^2+(z_2-y_2)^2+(z_3-y_3)^2]\\&=\frac{1}{2\times 3}\sum_{i=1}^{3}[(z_i-y_i)^2]\end{aligned}\tag{3}$$

w的梯度

我们用 J 的值作为基准，去求 w 对它的影响，也就是 J 对 w 的偏导数，就可以得到 w 的梯度了。从公式3看 J 的计算过程，z_1, z_2, z_3都对它有贡献；再从公式2看 z_1, z_2, z_3 的生成过程，都有 w 的参与。所以，J 对 w 的偏导应该是这样的：

$$\frac{\partial J}{\partial w} = \frac{\partial J}{\partial z_1}\frac{\partial z_1}{\partial w} + \frac{\partial J}{\partial z_2}\frac{\partial z_2}{\partial w} + \frac{\partial J}{\partial z_3}\frac{\partial z_3}{\partial w}$$

$$= \frac{1}{3}[(z_1 - y_1)x_1 + (z_2 - y_2)x_2 + (z_3 - y_3)x_3]$$

$$= \frac{1}{3}(x_1 \quad x_2 \quad x_3)\begin{pmatrix} z_1 - y_1 \\ z_2 - y_2 \\ z_3 - y_3 \end{pmatrix} \tag{4}$$

$$= \frac{1}{m}\sum_{i=1}^{m}(z_i - y_i)x_i$$

$$= \frac{1}{m}X^{\top} \cdot (Z - Y)$$

b的梯度

$$\frac{\partial J}{\partial b} = \frac{\partial J}{\partial z_1}\frac{\partial z_1}{\partial b} + \frac{\partial J}{\partial z_2}\frac{\partial z_2}{\partial b} + \frac{\partial J}{\partial z_3}\frac{\partial z_3}{\partial b}$$

$$= \frac{1}{3}[(z_1 - y_1) + (z_2 - y_2) + (z_3 - y_3)]$$

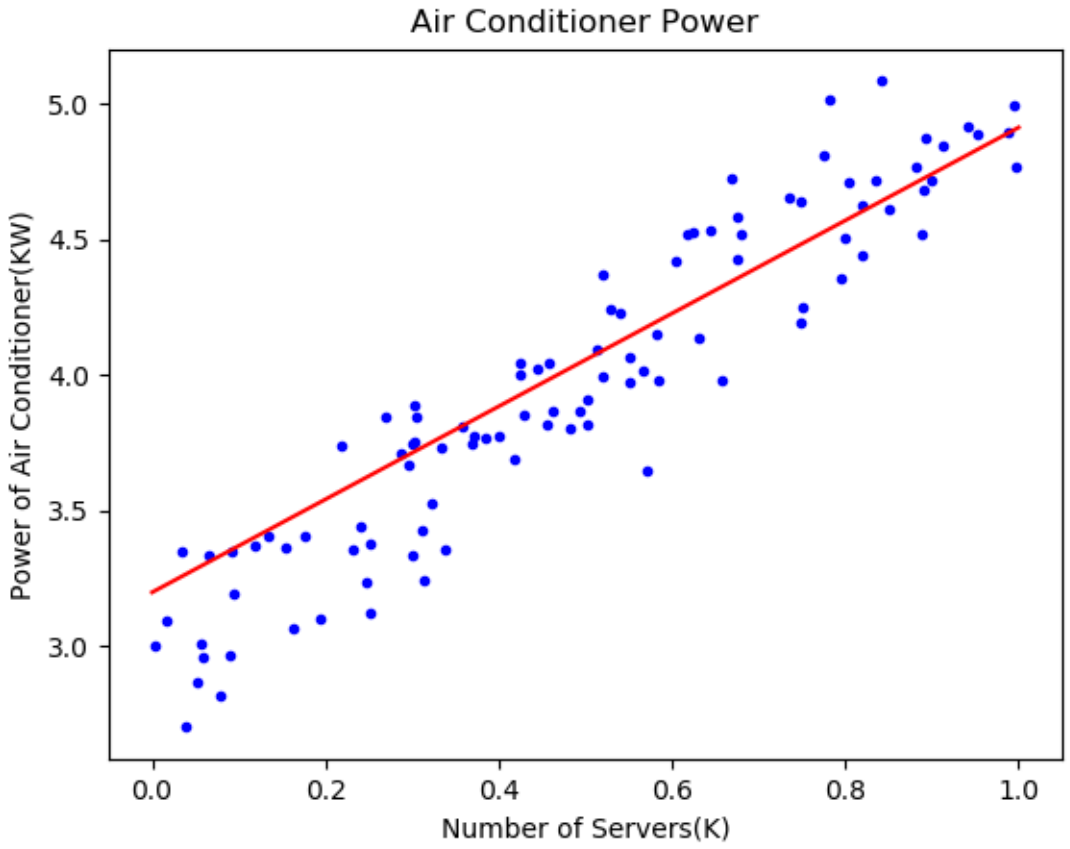
$$= \frac{1}{m}\sum_{i=1}^{m}(z_i - y_i)$$

$$= \frac{1}{m}(Z - Y)$$

梯度下降三种形式的比较

看看三种方式的运算结果

方法	w	b
最小二乘法	2.056827	2.965434
梯度下降法	1.71629006	3.19684087
神经网络法	1.71629006	3.19684087



神经网络的训练结果来看，拟合直线是斜着穿过样本点区域的，并没有在正中央的骨架上

神经网络不行?

初次使用神经网络，一定有水土不服的地方。最小二乘法可以得到数学解析解，所以它的结果是可信的。梯度下降法和神经网络法实际是一回事儿，只是梯度下降没有使用神经元模型而已。所以，接下来我们研究一下如何调整神经网络的训练过程，先从最简单的梯度下降的三种形式说起。

在下面的说明中，我们使用如下假设，以便简化问题易于理解：

1.使用可以解决本章的问题的线性回归模型，即 z=x·w+b；

2.样本特征值数量为1，即 x,w,b 都是标量；

3.使用均方差损失函数。

计算 w 的梯度：

$$\frac{\partial loss}{\partial w} = \frac{\partial loss}{\partial z_i}\frac{\partial z_i}{\partial w} = (z_i - y_i)x_i$$

计算 b 的梯度：

$$\frac{\partial loss}{\partial b} = \frac{\partial loss}{\partial z_i}\frac{\partial z_i}{\partial b} = z_i - y_i$$

神经网络不行?

初次使用神经网络，一定有水土不服的地方。最小二乘法可以得到数学解析解，所以它的结果是可信的。梯度下降法和神经网络法实际是一回事儿，只是梯度下降没有使用神经元模型而已。所以，接下来我们研究一下如何调整神经网络的训练过程，先从最简单的梯度下降的三种形式说起。

在下面的说明中，我们使用如下假设，以便简化问题易于理解：

1.使用可以解决本章的问题的线性回归模型，即 z=x·w+b；

2.样本特征值数量为1，即 x,w,b 都是标量；

3.使用均方差损失函数。

计算 w 的梯度：

$$\frac{\partial loss}{\partial w} = \frac{\partial loss}{\partial z_i}\frac{\partial z_i}{\partial w} = (z_i - y_i)x_i$$

计算 b 的梯度：

$$\frac{\partial loss}{\partial b} = \frac{\partial loss}{\partial z_i}\frac{\partial z_i}{\partial b} = z_i - y_i$$

单样本随机梯度法-SGD(Stochastic Gradient Descent)

SGD(Stochastic Gradient Descent)

样本访问示意图如图4-7所示。

图4-7 单样本访问方式

计算过程

假设一共100个样本，每次使用1个样本：

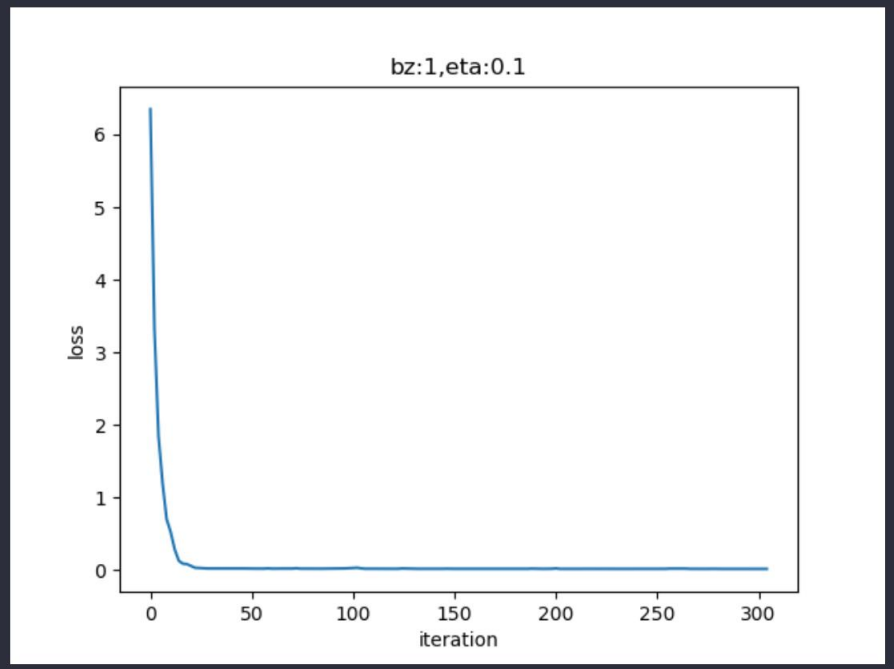
$$repeat\{ \quad for \quad i = 1,2,3,\ldots,100\{ \qquad z_i = x_i \cdot w + b \qquad dw = x_i \cdot (z_i - y_i) \qquad db = z_i - y_i \qquad w = w - \eta \cdot dw$$

特点

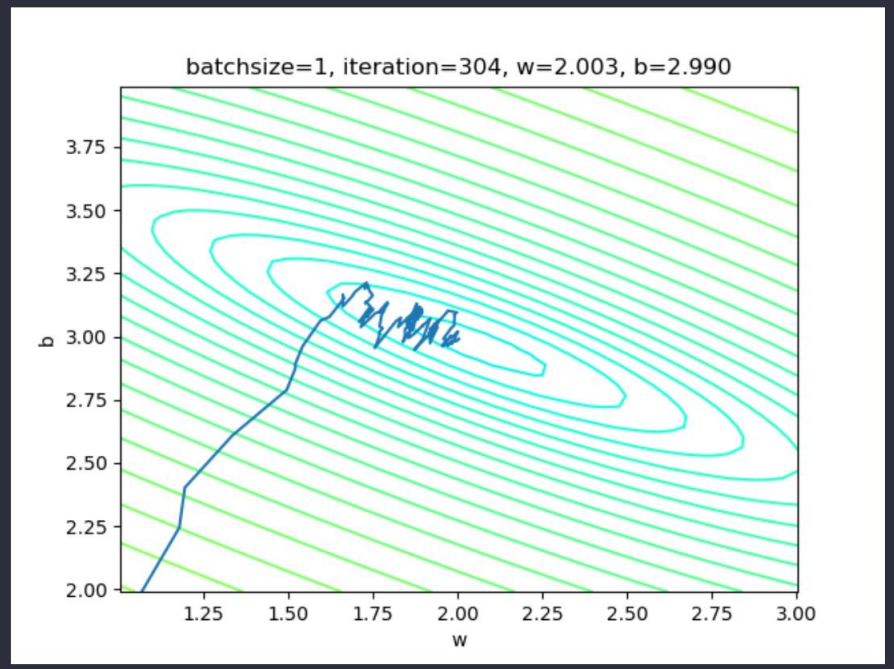
- 训练样本：每次使用一个样本数据进行一次训练，更新一次梯度，重复以上过程。
- 优点：训练开始时损失值下降很快，随机性大，找到最优解的可能性大。
- 缺点：受单个样本的影响最大，损失函数值波动大，到后期徘徊不前，在最优解附近震荡。不能并行计算。

单样本随机梯度法-SGD(Stochastic Gradient Descent)

损失函数值



梯度下降过程



左图，由于我们使用了限定的停止条件，即当损失函数值小于等于 0.02 时停止训练，所以，单样本方式迭代了300次后达到了精度要求。

右图是 w 和 b 共同构成的损失函数等高线图。梯度下降时，开始收敛较快，稍微有些弯曲地向中央地带靠近。到后期波动较大，找不到准确的前进方向，曲折地达到中心附近。

小批量样本梯度下降-Mini-Batch Gradient Descent

图4-8 小批量样本访问方式

计算过程

假设一共100个样本，每个小批量5个样本：

$$repeat\{ \quad for \quad i=1,6,11,\ldots,96\{ \qquad z_i=x_i\cdot w+b \qquad z_{i+1}=x_{i+1}\cdot w+b \qquad \ldots \qquad z_{i+4}=x_{i+4}\cdot w+b \qquad dw$$

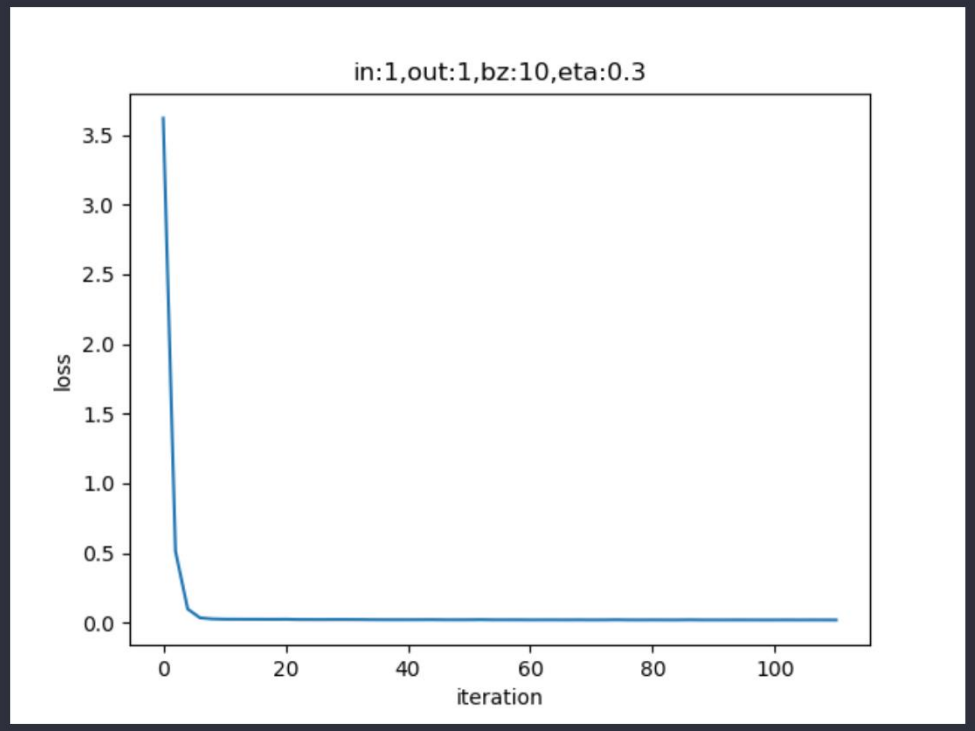
上述算法中，循环体中的前5行分别计算了 $z_i, z_{i+1}, \ldots, z_{i+4}$，可以换成一次性的矩阵运算。

特点

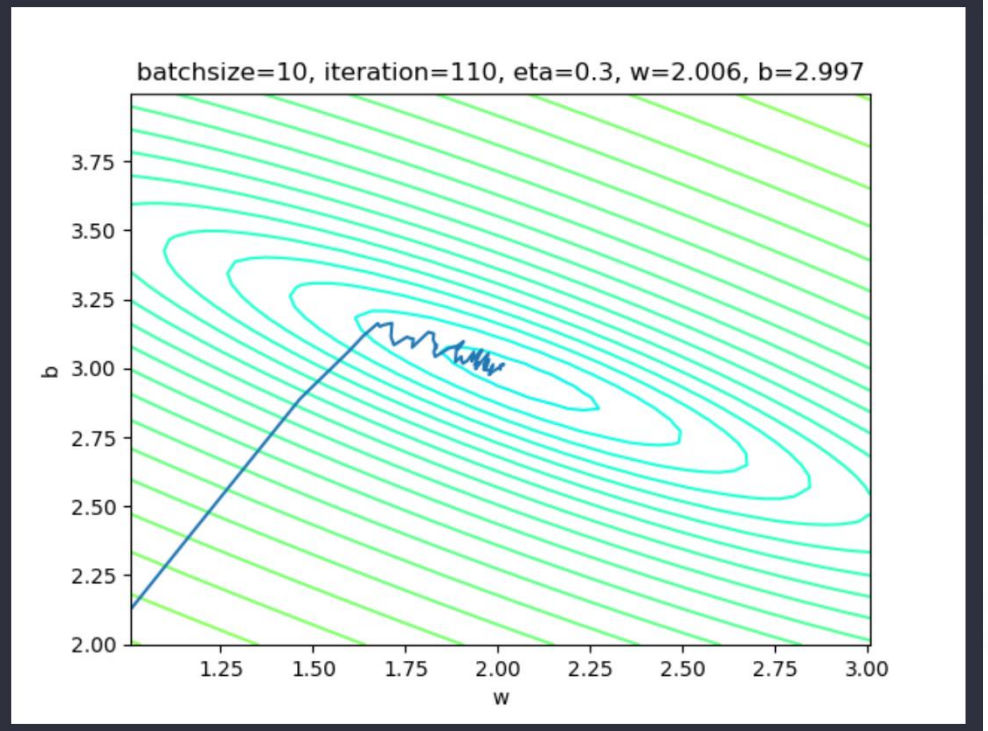
- 训练样本：选择一小部分样本进行训练，更新一次梯度，然后再选取另外一小部分样本进行训练，再更新一次梯度。
- 优点：不受单样本噪声影响，训练速度较快。
- 缺点：batch size的数值选择很关键，会影响训练结果。

小批量样本梯度下降-Mini-Batch Gradient Descent

损失函数值



梯度下降过程



梯度下降时，在接近中心时有小波动

小批量样本梯度下降-Mini-Batch Gradient Descent

小批量的大小通常由以下几个因素决定：

更大的批量会计算更精确的梯度，但是回报却是小于线性的。

极小批量通常难以充分利用多核架构。这决定了最小批量的数值，低于这个值的小批量处理不会减少计算时间。

如果批量处理中的所有样本可以并行地处理，那么内存消耗和批量大小成正比。对于多硬件设施，这是批量大小的限制因素。

某些硬件上使用特定大小的数组时，运行时间会更少，尤其是GPU，通常使用2的幂数作为批量大小可以更快，如32,64,128,256，大模型时尝试用16。

可能是由于小批量在学习过程中加入了噪声，会带来一些正则化的效果。泛化误差通常在批量大小为1时最好。因为梯度估计的高方差，小批量使用较小的学习率，以保持稳定性，但是降低学习率会使迭代次数增加。

在实际工程中，我们通常使用小批量梯度下降形式

全批量样本梯度下降-Full Batch Gradient Descent

图4-9 全批量样本访问方式

计算过程

假设一共100个样本，每次使用全部样本：

$$repeat\{ \quad z_1 = x_1 \cdot w + b \quad z_2 = x_2 \cdot w + b \quad \dots \quad z_{100} = x_{100} \cdot w + b \quad dw = \frac{1}{100}\sum_{i=1}^{100} x_i \cdot (z_i - y_i) \quad db = \frac{1}{100}\sum_{i=1}^{100}(z$$

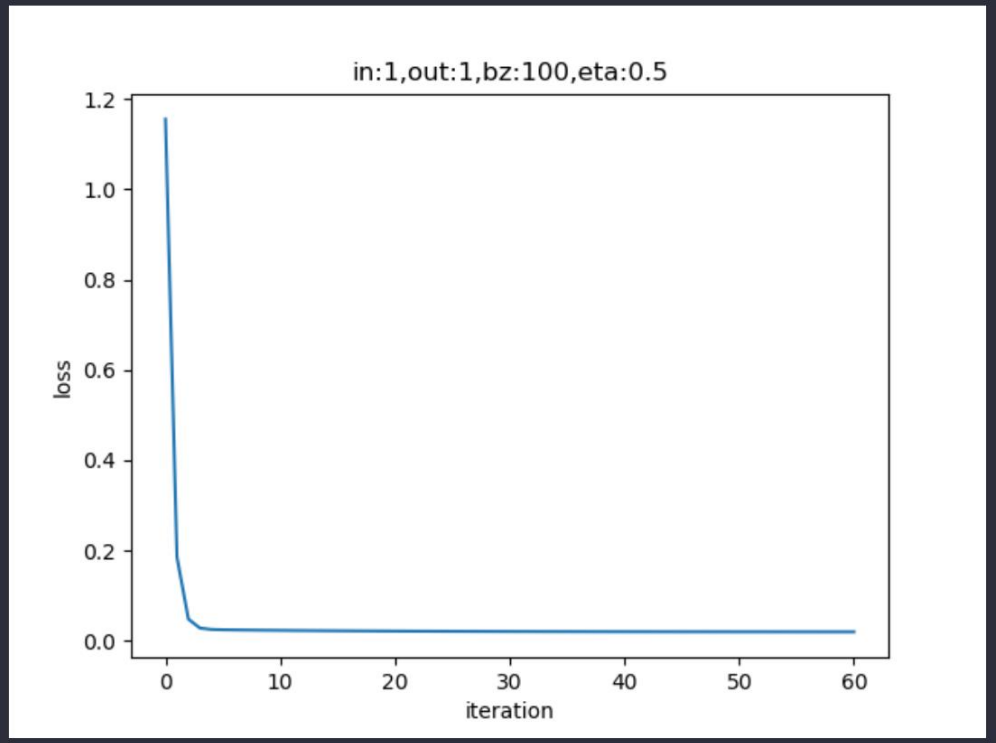
上述算法中，循环体中的前100行分别计算了 $z_1, z_2, \dots, z_{100}$，可以换成一次性的矩阵运算。

特点

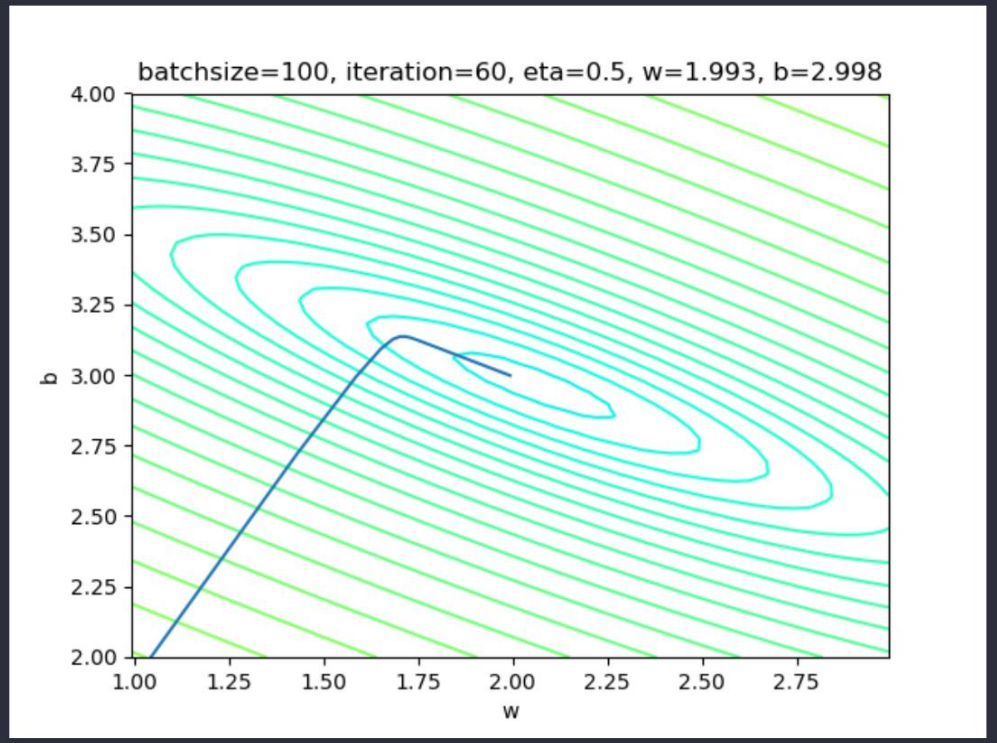
- 训练样本：每次使用全部数据集进行一次训练，更新一次梯度，重复以上过程。
- 优点：受单个样本的影响最小，一次计算全体样本速度快，损失函数值没有波动，到达最优点平稳。方便并行计算。
- 缺点：数据量较大时不能实现（内存限制），训练过程变慢。初始值不同，可能导致获得局部最优解，并非全局最优解。

全批量样本梯度下降-Full Batch Gradient Descent

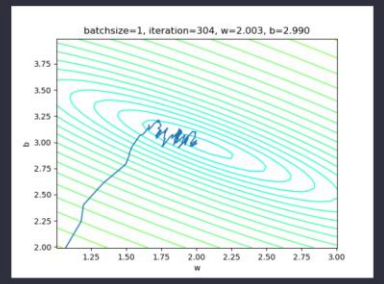
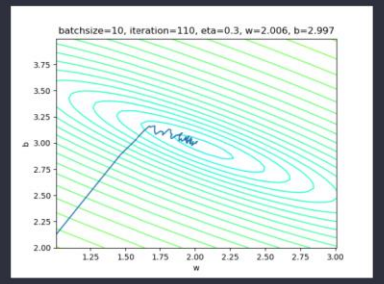
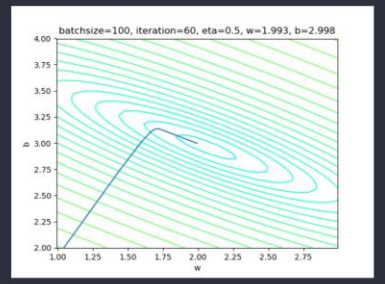
损失函数值

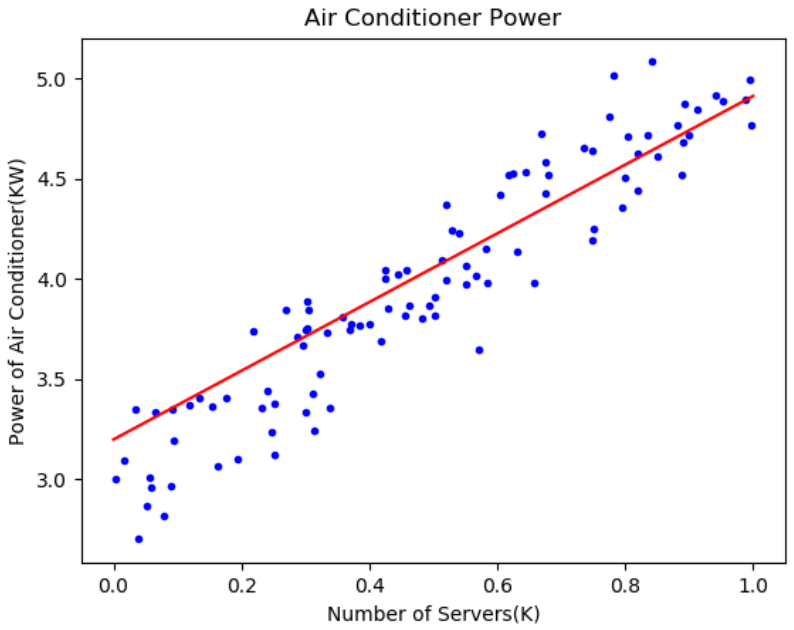


梯度下降过程



三种方式比较

	单样本	小批量	全批量
梯度下降过程图解			
批大小	1	10	100
学习率	0.1	0.3	0.5
迭代次数	304	110	60
epoch	3	10	60
结果	w=2.003, b=2.990	w=2.006, b=2.997	w=1.993, b=2.998



五. 示例 – 预测上海天气

六. 小结

Microsoft

Reactor

Thank You!