Microsoft

Reactor

一起学人工智能系列
- 线性回归2

2021-09-07

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

回顧

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。

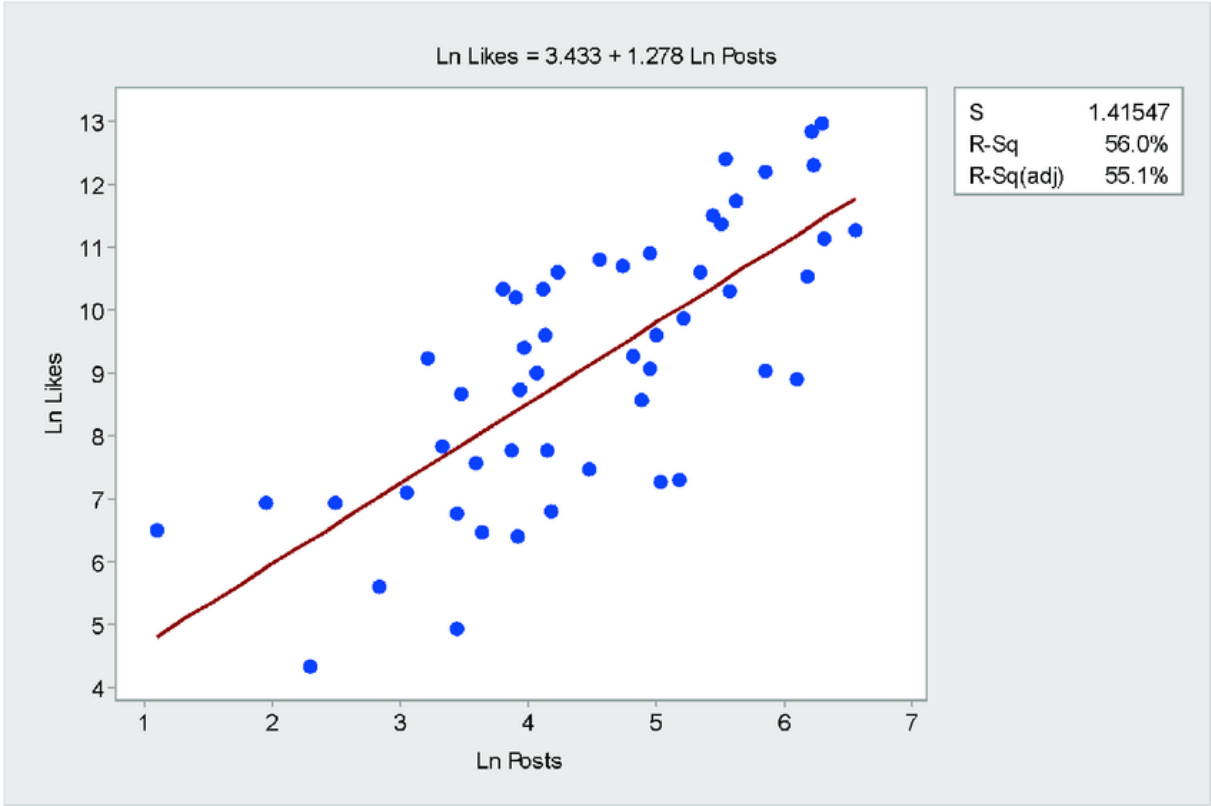
④利用所求的关系式对某一过程进行预测或控制。

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。

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

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：

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

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

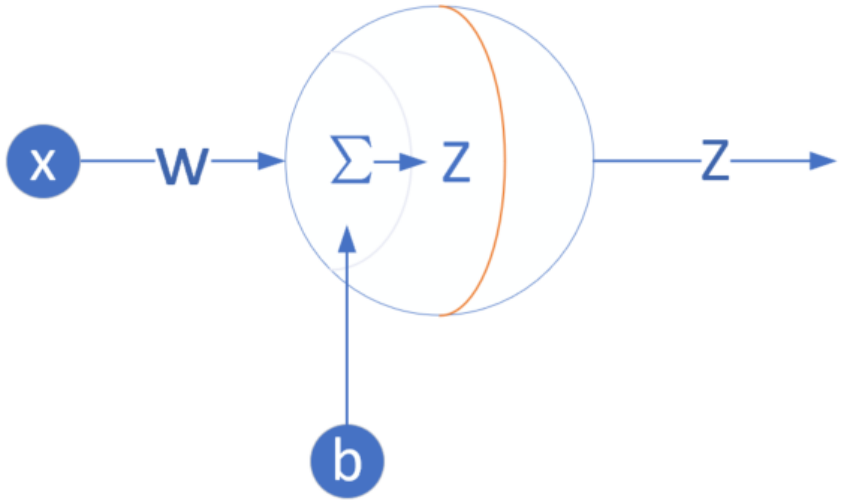
$$\frac{\partial loss}{\partial w} = \frac{\partial loss}{\partial z_i}\frac{\partial z_i}{\partial w} = (z_i - y_i)x_i$$

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



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。

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

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。

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。

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：

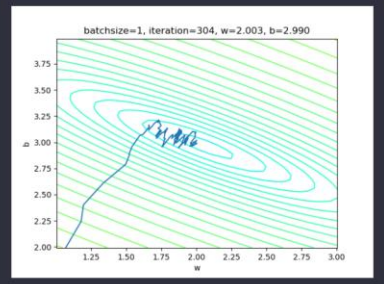
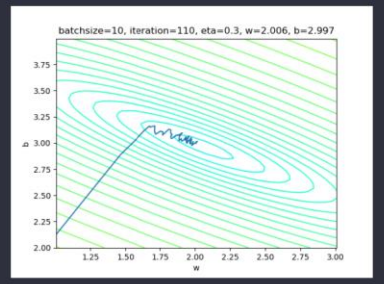
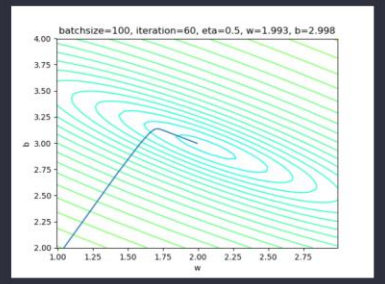
$$repeat\{ \quad z_1 = x_1 \cdot w + b \quad z_2 = x_2 \cdot w + b \quad \dots \quad z_{100} = x_{100} \cdot w + b \quad dw = \frac{1}{100}\sum_{i=1}^{100} x_i \cdot (z_i - y_i) \quad db = \frac{1}{100}\sum_{i=1}^{100}(z$$

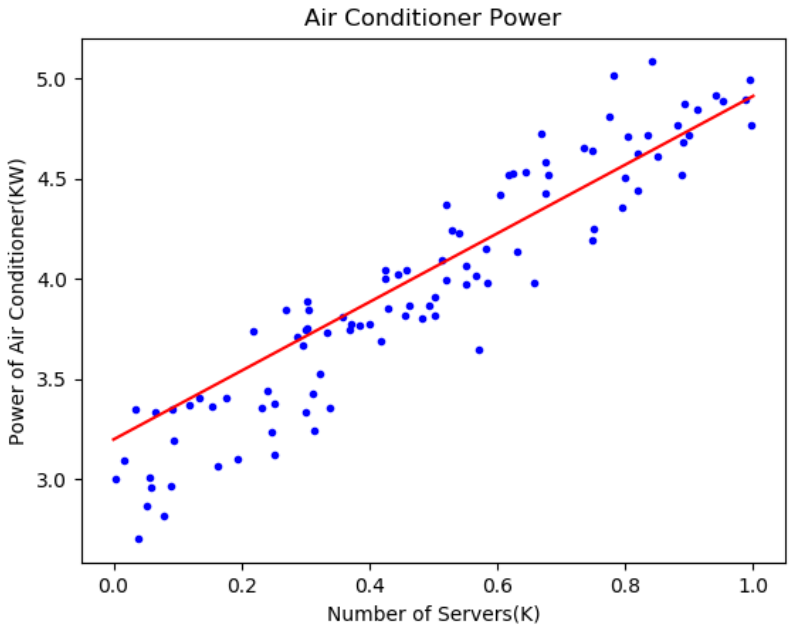
上述算法中，循环体中的前100行分别计算了 $z_1, z_2, \dots, z_{100}$，可以换成一次性的矩阵运算。

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。

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



一．多变量线性回归

从北京通州房价谈起

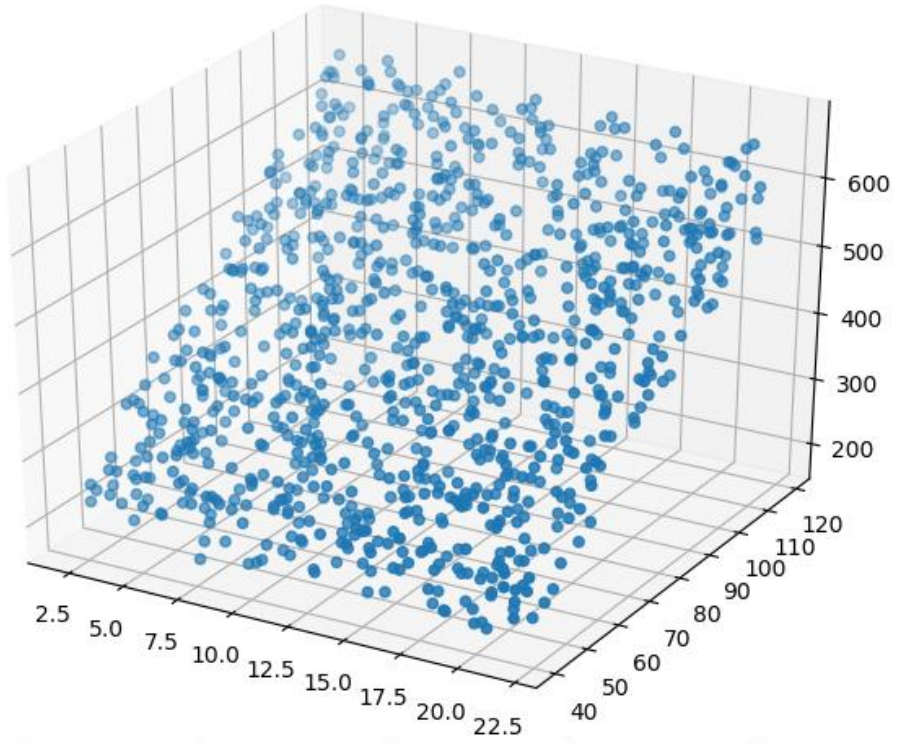
样本序号	地理位置	居住面积	价格（万元）
1	10.06	60	302.86
2	15.47	74	393.04
3	18.66	46	270.67
4	5.20	77	450.59
...	...	...	...

- 特征值1 - 地理位置，统计得到：
- 最大值：21.96公里
- 最小值：2.02公里
- 平均值：12.13公里

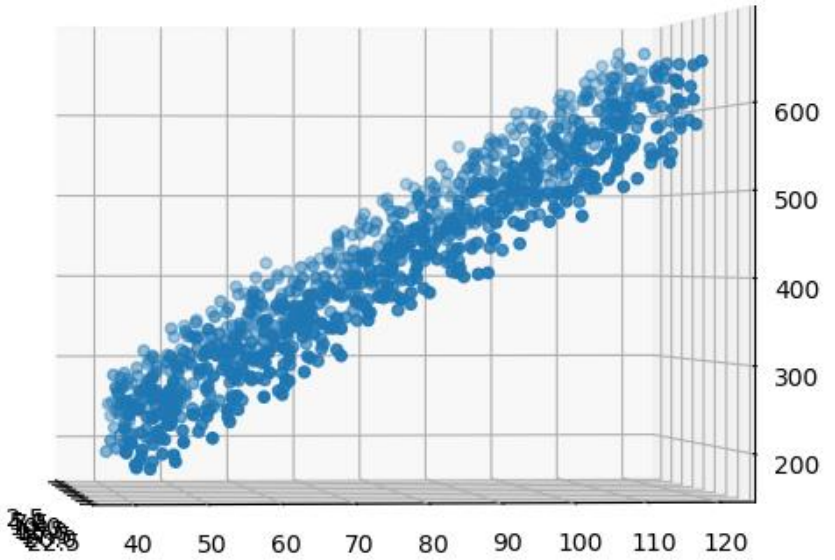
- 特征值2 - 房屋面积，统计得到：
- 最大值：119平米
- 最小值：40平米
- 平均值：78.9平米

- 标签值 - 房价，单位为百万元：
- 最大值：674.37
- 最小值：181.38
- 平均值：420.64

从北京通州房价谈起



正向



侧向

从北京通州房价谈起

预测15公里、93平米?

多元线性回归

线性回归问题，而且是典型的多元线性回归，即包括两个或两个以上自变量的回归。多元线性回归的函数模型如下：

$$y = a_0 + a_1 x_1 + a_2 x_2 + \cdots + a_k x_k$$

具体化到房价预测问题，上面的公式可以简化成：

$$z = x_1 \cdot w_1 + x_2 \cdot w_2 + b$$

多元线性回归

1.自变量对因变量必须有显著的影响，并呈密切的线性相关；

2.自变量与因变量之间的线性相关必须是真实的，而不是形式上的；

3.自变量之间应具有一定的互斥性，即自变量之间的相关程度不应高于自变量与因变量之因的相关程度；

4.自变量应具有完整的统计数据，其预测值容易确定。

数学方式解决问题

多元线性方程式

$$y = a_0 + a_1x_1 + a_2x_2 + \cdots + a_kx_k$$

函数拟合（回归）时，我们假设函数 为

$$H(w,b) = b + x_1w_1 + x_2w_2 + \cdots + x_nw_n$$

b = w0

$$H(W) = w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 + \cdots + x_n \cdot w_n$$

x 是一个样本的 n 个特征值，如果我们把 m 个样本一起计算，将会得到下面这个矩阵

$$H(W) = X \cdot W$$

数学方式解决问题

$$X = \begin{pmatrix} 1 & x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ 1 & x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{pmatrix} * W = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix} = Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

$$H(W) = X \cdot W = Y$$

$$X^\top XW = X^\top Y$$

其中，$X^\top$ 是 X 的转置矩阵，$ X^{\top}X$ 一定是个方阵，并且假设其存在逆矩阵，把它移到等式右侧来：

$$W = (X^\top X)^{-1} X^\top Y$$

数学方式解决问题

我们仍然使用均方差损失函数（略去了系数$\frac{1}{2m}$）：

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

把b看作是一个恒等于1的feature，并把$Z=XW$计算公式带入，并变成矩阵形式：

$$J(W)=\sum_{i=1}^{m}\left(\sum_{j=0}^{n}x_{ij}w_j-y_i\right)^2=(XW-Y)^\top\cdot(XW-Y)$$

对W求导，令导数为0，可得到W的最小值解：

$$\frac{\partial J(W)}{\partial W}=\frac{\partial}{\partial W}[(XW-Y)^\top\cdot(XW-Y)]$$

$$=\frac{\partial}{\partial W}[(W^\top X^\top-Y^\top)\cdot(XW-Y)]$$

$$=\frac{\partial}{\partial W}[(W^\top X^\top XW-W^\top X^\top Y-Y^\top XW+Y^\top Y)]$$

求导后（请参考矩阵/向量求导公式）：

第一项的结果是：$2X^\top XW$（分母布局，denominator layout）

第二项的结果是：$X^\top Y$（分母布局方式，denominator layout）

第三项的结果是：$X^\top Y$（分子布局方式，numerator layout，需要转置$Y^\top X$）

第四项的结果是：0

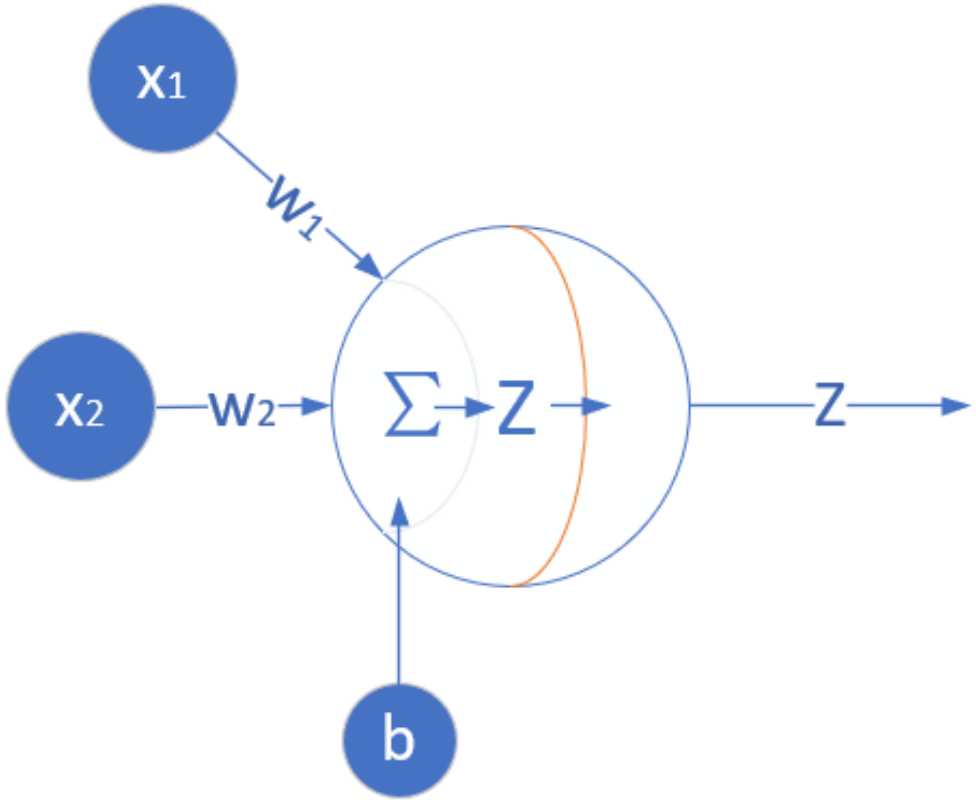
再令导数为0：

$$X^\top XW=X^\top Y$$

其中，$X^\top$是X的转置矩阵，$ X^{\top}X$ 一定是个方阵，并且假设其存在逆矩阵，把它移到等式右侧来：

$$W=(X^\top X)^{-1}X^\top Y$$

神经网络解法



1.没有中间层，只有输入项和输出层（输入项不算做一层）；

2.输出层只有一个神经元；

3.神经元有一个线性输出，不经过激活函数处理，即在下图中，经过 Σ 求和得到 Z 值之后，直接把 Z 值输出。

神经网络解法

输入层

单独看第一个样本是这样的：

$$x_1 = (x_{11} \quad x_{12}) = (10.06 \quad 60)$$

$$y_1 = (302.86)$$

一共有1000个样本，每个样本2个特征值，X就是一个1000×2的矩阵：

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{1000} \end{pmatrix} = \begin{pmatrix} x_{1,1} & x_{1,2} \\ x_{2,1} & x_{2,2} \\ \vdots & \vdots \\ x_{1000,1} & x_{1000,2} \end{pmatrix}$$

$$Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{1000} \end{pmatrix} = \begin{pmatrix} 302.86 \\ 393.04 \\ \vdots \\ 450.59 \end{pmatrix}$$

神经网络解法

权重 W 和 B

由于输入层是两个特征，输出层是一个变量，所以 W 的形状是 2×1，而 B 的形状是 1×1。

$$W = \begin{pmatrix} w_1 \\ \\ w_2 \end{pmatrix}$$

$$B = (b)$$

B 是个单值，因为输出层只有一个神经元，所以只有一个bias，每个神经元对应一个bias，如果有多个神经元，它们都会有各自的b值。

输出层

由于我们只想完成一个回归（拟合）任务，所以输出层只有一个神经元。由于是线性的，所以没有用激活函数。 $$ \begin{aligned} Z&= \begin{pmatrix} x_{11} & x_{12} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix} +(b) \\ &=x_{11}w_1+x_{12}w_2+b \end{aligned} $$

写成矩阵形式：

$$Z = X \cdot W + B$$

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。

$$loss_i(W, B) = \frac{1}{2}(z_i - y_i)^2 \tag{1}$$

二．样本特征数据标准化

样本特征数据标准化

数据标准化（Normalization），又可以叫做数据归一化。

为什么数据标准化

神经网络是以样本在事件中的统计分布概率为基础进行训练和预测的，所以它对样本数据的要求比较苛刻。具体说明如下：

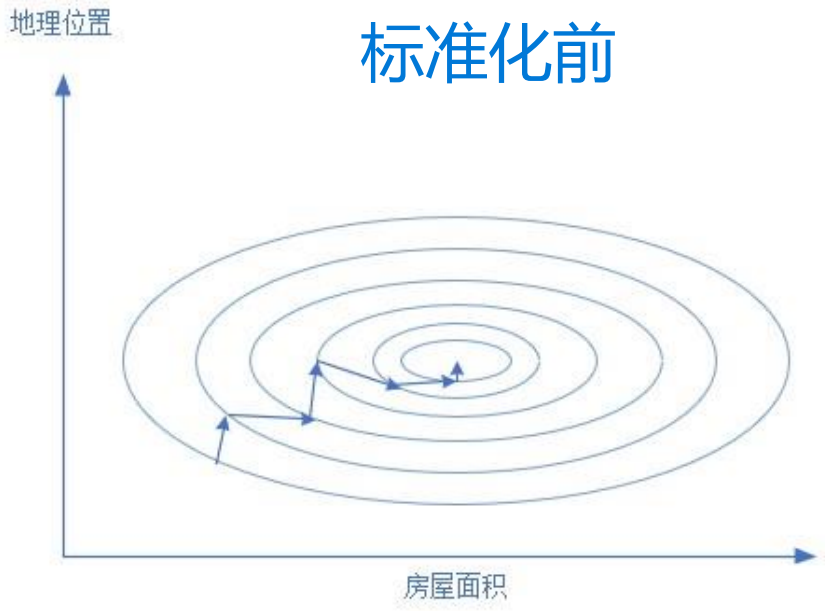
1.样本的各个特征的取值要符合概率分布，即 [0,1]。
2.样本的度量单位要相同。我们并没有办法去比较1米和1公斤的区别，但是，如果我们知道了1米在整个样本中的大小比例，以及1公斤在整个样本中的大小比例，比如一个处于0.2的比例位置，另一个处于0.3的比例位置，就可以说这个样本的1米比1公斤要小。
3.神经网络假设所有的输入输出数据都是标准差为1，均值为0，包括权重值的初始化，激活函数的选择，以及优化算法的设计。
4.数值问题
5.标准化可以避免一些不必要的数值问题。因为激活函数sigmoid/tanh的非线性区间大约在 ，[−1.7，1.7]。意味着要使神经元有效，线性计算输出的值的数量级应该在1（1.7所在的数量级）左右。这时如果输入较大，就意味着权值必须较小，一个较大，一个较小，两者相乘，就引起数值问题了。
6.梯度更新
7.若果输出层的数量级很大，会引起损失函数的数量级很大，这样做反向传播时的梯度也就很大，这时会给梯度的更新带来数值问题。
8.学习率
9.如果梯度非常大，学习率就必须非常小，因此，学习率（学习率初始值）的选择需要参考输入的范围，不如直接将数据标准化，这样学习率就不必再根据数据范围作调整。对 w1 适合的学习率，可能相对于 w2 来说会太小，若果使用适合 w1 的学习率，会导致在 w2 方向上步进非常慢，从而消耗非常多的时间；而使用适合 w2 的学习率，对 w1 来说又太大，搜索不到适合 w1 的解。

为什么数据标准化

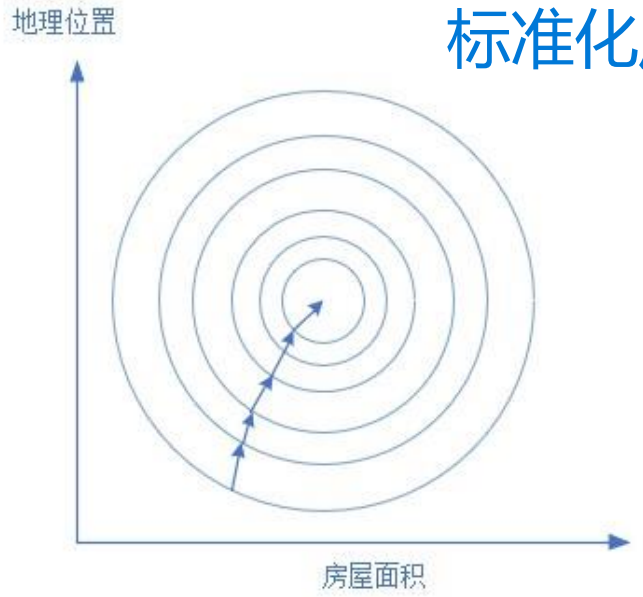
在房价数据中，地理位置的取值范围是 [2,20]，而房屋面积的取值范围为 [40,120]，二者相差太远，放在一起计算会怎么样?

根据公式z=x1w1+x2w2+b，神经网络想学习 w1 和 w2，但是数值范围问题导致神经网络来说很难“理解”。

标准化前



标准化后



归一化

把数据线性地编程[0,1]/[-1,1]之间的小数，把带单位的数据(米，千克)变成无量纲的数据。区间缩放。

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Min-Max归一化

$$x_{new} = \frac{x - \bar{x}}{x_{max} - x_{min}}$$

平均值归一化

对数转换

$$y = \log(x)$$

反余切对换

$$y = \operatorname{arccot}(x) \cdot 2/\pi$$

非线性归一化

标准化

把每个特征值中的所有数据，变成平均值为0，标准差为1的数据，最后为正态分布。Z-score规范化也叫标准差标准化或零均值标准化，其中std是标准差。

$$x_{new} = \frac{x_i - \bar{x}}{std}$$

中心化

平均值为0，无标准差要求。

$$x_{new} = x_i - \bar{x}$$

示例

样本序号	地理位置	居住面积	价格（万元）
1	10.06	60	302.86
2	15.47	74	393.04
3	18.66	46	270.67
4	5.20	77	450.59
...	...	...	...

标准化后

样本序号	地理位置	居住面积	价格（万元）
1	0.4033	0.2531	302.86
2	0.6744	0.4303	393.04
3	0.8341	0.0759	270.67
4	0.1592	0.4683	450.59
...	...	...	...

五. 小结

Microsoft

Reactor

Thank You!