# 智能之门

## 神经网络和深度学习入门

（基于PYTHON的实现）

STEP 2 线性回归

# 第 5 章

## 多入单出的单层神经网络

5.1 多变量线性回归问题

5.2 正规方程法

5.3 神经网络法

5.4 样本特征数据标准化

5.5 还原参数值

5.6 标签值标准化

当接收多个变量输入时，成为多变量线性回归问题。此时要通过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。

# 5.1 多变量线性回归问题

问题：在北京通州，距离通州区中心15公里的一套93平米的房子，大概是多少钱？

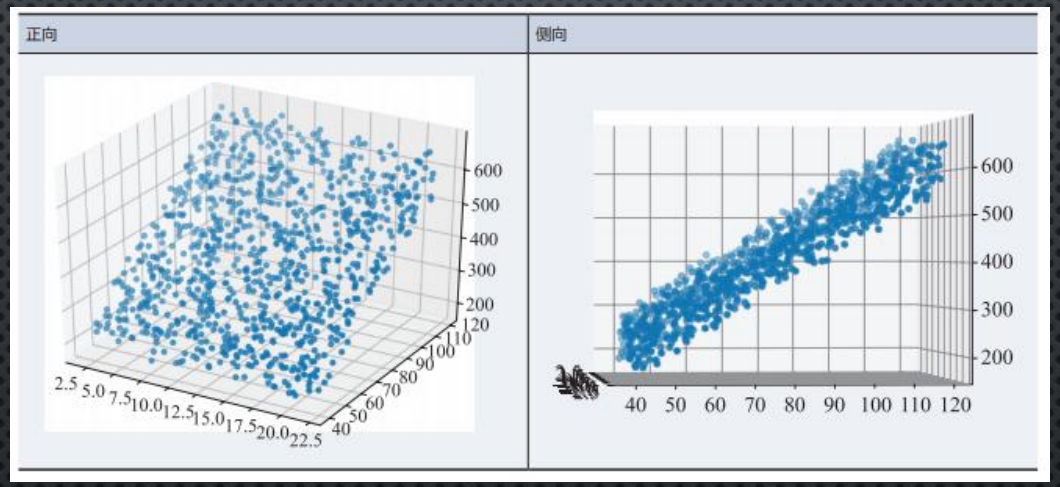
房价预测问题是机器学习的一个入门话题，著名的波士顿的房价数据及相关的比赛已经很多了，但是美国的房子都是独栋的，前院后院停车库游泳池等等参数非常多，初学者可能理解起来有困难。我们不妨用简化版的北京通州的房价来举例，感受一下房价预测的过程。

影响北京通州房价的因素有很多，居住面积、地理位置、朝向、学区房、周边设施、建筑年份等等，其中，面积和地理位置是两个比较重要的因素。地理位置信息一般采用经纬度方式表示，但是经纬度是两个特征值，联合起来才有意义，因此，我们把它转换成了到通州区中心的距离。

| 样本序号 | 地理位置/km | 居住面积/$m^2$ | 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 |
| … | … | … | … |

# 5.1 多变量线性回归问题

这个数据是三维的，可以用两个特征值作为$x$和$y$，用标签值作为$z$，在三维空间中展示如下：




从正向看，点集很像一块草坪，似乎是一个平面。再从侧向看，点集又和上一章中的直线拟合数据很像。所以，对于这种三维的线性拟合，我们可以把它想象成为拟合一个平面，这个平面会位于这块“草坪”的中位，把“草坪”分割成上下两块更薄的“草坪”，最终使得所有样本点到这个平面的距离的平方和最小。

# 5.1 多变量线性回归问题

## ➢ 多元线性回归模型

- 多元线性回归的函数模型如下：

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

- 对于一般的应用问题，建立多元线性回归模型时，为了保证回归模型具有优良的解释能力和预测效果，应首先注意自变量的选择，其准则是：
  - ✓ 自变量对因变量必须有显著的影响，并呈密切的线性相关；
  - ✓ 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；
  - ✓ 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；
  - ✓ 自变量应具有完整的统计数据，其预测值容易确定。

# 5.1 多变量线性回归问题

- 求解方法
  - 如果用传统的数学方法解决这个问题，我们可以使用正规方程法，从而可以得到数学解析解；然后再使用神经网络方式来求得近似解，从而比较两者的精度，再进一步调试神经网络的参数，达到学习的目的。
  - 两种方法的对比如下：

| 方法 | 解方程法 | 梯度下降法 |
| --- | --- | --- |
| 原理 | 几次矩阵运算 | 多次迭代 |
| 特殊要求 | $X^TX$ 的逆矩阵存在 | 需要确定学习率 |
| 复杂度 | $O(n^3)$ | $O(n^2)$ |
| 适用样本数 | $m<10\,000$ | $m\geqslant 10\,000$ |

# 5.2 正规方程法

➢ 简单的数学推导

- 假设拟合函数为

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

- 采用多样本计算，假设函数的输出与真实值一致，得到矩阵形式：$H = XW = Y$，其中：

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

- 等式两边同时乘以 $X^T$，得到正规方程：$X^T XW = X^T Y$。
- 若方阵 $X^T X$ 可逆，移项即可得到正规方程解：

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

# 5.2 正规方程法

- 复杂的数学推导
  - 仍然采用均方差损失函数（省略了系数）：

$$J(W) = \sum_{i=1}^{m} (z_i - y_i)^2 = (XW - Y)^T (XW - Y)$$

  - 对 $W$ 求导并置之为 0，得到：

$$\frac{\partial J(W)}{\partial W} = \frac{\partial}{\partial W}[W^T X^T XW - W^T X^T Y - Y^T XW + Y^T Y] = 2X^T XW - 2X^T Y = 0$$

  - 因此同样可以得到正规方程： $X^T XW = X^T Y$。
  - 若方阵 $X^T X$ 可逆，移项即可得到正规方程解：

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

# 5.2 正规方程法

- 方针$X^TX$不可逆的原因：矩阵$X$不是列满秩的。
  - 特征值冗余，比如正方形的边长与面积的关系不能作为两个特征同时存在；
  - 特征数量过多，甚至超过了样本数量。

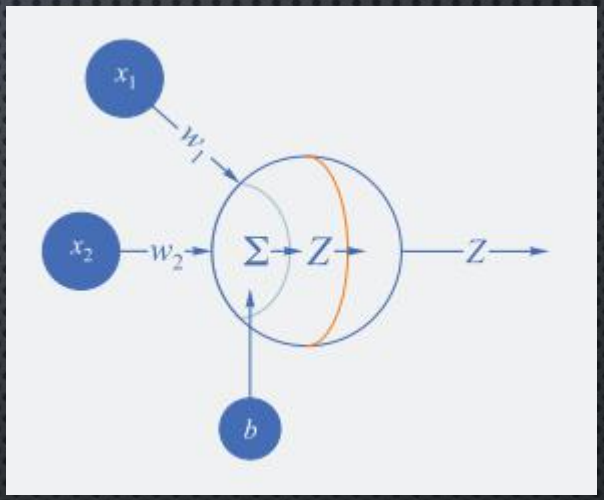
前面所给的例子中不存在这一问题，因而可以用正规方程解法直接求得数学解析解。

正规方程解得到的房价预测值约为486万元。

# 5.3 神经网络法

定义如图所示的单层神经网络，该神经网络的特点如下：

- ✓ 没有中间层，只有输入项和输出层（输入项不算一层）。
- ✓ 输出层只有一个神经元。
- ✓ 神经元有一个线性输出，不经过激活函数处理。
- ✓ 可以同时接收多个输入，这是神经网络能够处理复杂逻辑的根本原因。



# 5.3 神经网络法

- 输入层
  - 一共有1000个样本，每个样本2个特征值。
- 权重$W$和$B$
  - $W$为$2\times1$矩阵，$B$为$1\times1$矩阵。
- 输出层
  - 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。
- 损失函数
  - 因为是线性回归问题，所以损失函数继续使用均方差函数。

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

# 5.3 神经网络法

- 反向传播
  - 单样本多特征计算
    - ✓ 本部分为多特征值公式：

$$z_i = x_{i1}w_1 + x_{i2}w_2 + b$$

    - ✓ 对参数单独求导后整合：

$$\frac{\partial loss_i}{\partial w_1} = \frac{\partial loss_i}{\partial z_i} \cdot \frac{\partial z_i}{\partial w_1} = (z_i - y_i) \cdot x_{i1}, \qquad \frac{\partial loss_i}{\partial w_2} = \frac{\partial loss_i}{\partial z_i} \cdot \frac{\partial z_i}{\partial w_2} = (z_i - y_i) \cdot x_{i2}$$

$$\frac{\partial loss_i}{\partial W} = \begin{pmatrix} \frac{\partial loss_i}{\partial w_1} \\ \frac{\partial loss_i}{\partial w_2} \end{pmatrix} = \begin{pmatrix} x_{i1} \\ x_{i2} \end{pmatrix} \cdot (z_i - y_i) = x_i^T (z_i - y_i), \qquad \frac{\partial loss_i}{\partial B} = \left( \frac{\partial loss_i}{\partial b} \right) = (z_i - y_i)$$

# 5.3 神经网络法

- 多样本多特征计算
  - ✓ 以 $m=3$ 的情形作实例化推导：

$$J(W,B)=\frac{1}{2\times 3}[(z_1-y_1)^2+(z_2-y_2)^2+(z_3-y_3)^2]$$

$$\frac{\partial J}{\partial W}=\begin{pmatrix}\frac{\partial J}{\partial w_1}\\ \frac{\partial J}{\partial w_2}\end{pmatrix}=\begin{pmatrix}\frac{\partial J}{\partial z_1}\cdot\frac{\partial z_1}{\partial w_1}+\frac{\partial J}{\partial z_2}\cdot\frac{\partial z_2}{\partial w_1}+\frac{\partial J}{\partial z_3}\cdot\frac{\partial z_3}{\partial w_1}\\ \frac{\partial J}{\partial z_1}\cdot\frac{\partial z_1}{\partial w_2}+\frac{\partial J}{\partial z_2}\cdot\frac{\partial z_2}{\partial w_2}+\frac{\partial J}{\partial z_3}\cdot\frac{\partial z_3}{\partial w_2}\end{pmatrix}=\begin{pmatrix}\frac{\partial z_1}{\partial w_1} & \frac{\partial z_2}{\partial w_1} & \frac{\partial z_3}{\partial w_1}\\ \frac{\partial z_1}{\partial w_2} & \frac{\partial z_2}{\partial w_2} & \frac{\partial z_3}{\partial w_2}\end{pmatrix}\begin{pmatrix}\frac{\partial J}{\partial z_1}\\ \frac{\partial J}{\partial z_2}\\ \frac{\partial J}{\partial z_3}\end{pmatrix}=\frac{1}{3}X^T(Z-Y)$$

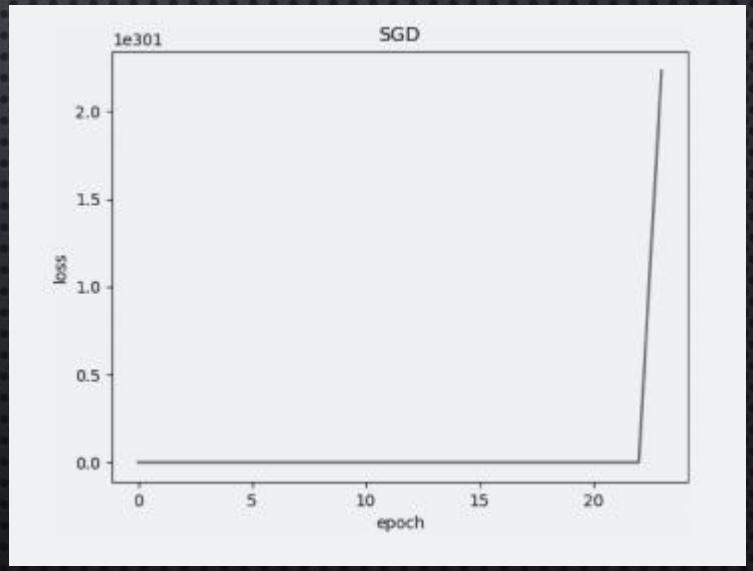
  - ✓ 事实上有以下等式成立：

$$\frac{\partial J}{\partial W}=\frac{1}{m}X^T(Z-Y),\qquad \frac{\partial J}{\partial B}=\frac{1}{m}(Z-Y)$$

# 5.3 神经网络法

- 代码运行结果
  - 采用SGD，前 10 次迭代后损失函数值已经达到了 6.83e+66，而且越往后运行值越大，最后终于溢出了。下图所示的损失函数历史记录也表明了这一过程。

这可能就是传说中的梯度爆炸！数值太大，导致计算溢出了。不过不用担心，这是我们第一次遇到这个情况，但也相信这不会是最后一次，毕竟这种情况在神经网络中太常见了。



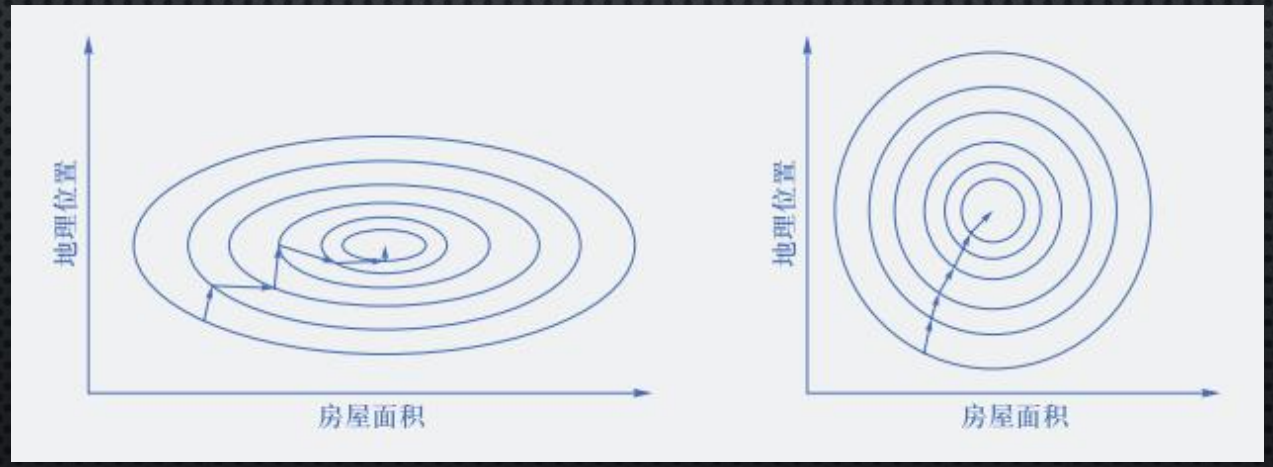
# 5.4 样本特征数据标准化

- 数据标准化/归一化：深度学习的必要步骤之一，已经是大师们的必杀技能。理论层面上，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。
  - 样本的各个特征的取值要符合概率分布，即[0,1]。
  - 样本的度量单位要相同。
  - 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。
  - 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左右。
  - 若输出层的数量级很大，损失函数的数量级会很大，这样做反向传播时的梯度也就很大，给梯度的更新带来数值问题。
  - 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。

# 5.4 样本特征数据标准化

- 损失函数等高线图展示标准化的必要性

数值范围问题导致神经网络来说很难“理解”。下图展示了标准化前后的情况损失函数值的等高图，意思是地理位置和房屋面积取不同的值时，作为组合来计算损失函数值时，形成的类似地图的等高图，左侧为标准化前，右侧为标准化后。房屋面积的取值范围是[40,120]，而地理位置的取值范围是[2,20]，二者会形成一个很扁的椭圆，如左侧。这样在寻找最优解的时候，过程会非常曲折。运气不好的话，根本就没法训练。



# 5.4 样本特征数据标准化

➢ 常用方法

- 归一化：把数据线性映射至 [0,1] 或 [−1,1] 之间，把带单位的数据变成无量纲的数据，区间缩放。
  - ✓ Min-Max 归一化

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

  - ✓ 平均值归一化

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

  - ✓ 非线性归一化（对数归一化、反正切归一化等）
  - ✓ 比例法：要求数据非负且其和为1。

$$x_{new} = \frac{x_k}{\sum_{i=1}^{m} x_i}$$

- 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。（Z-Score 规范化，std 为标准差）

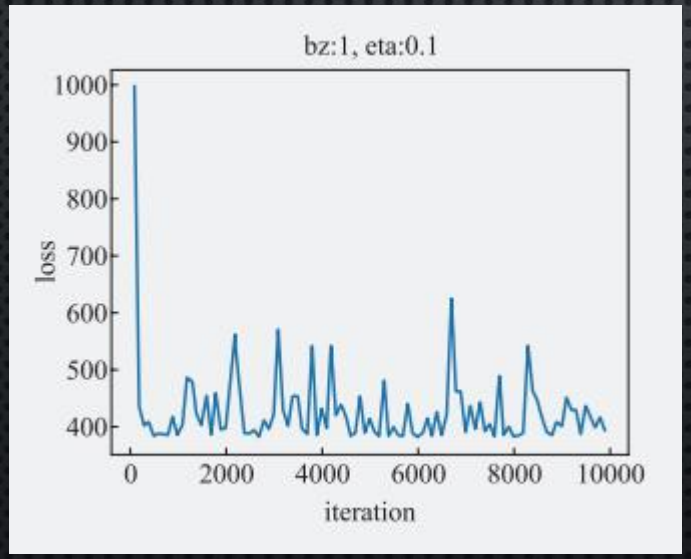
$$x_{new} = \frac{x - \bar{x}}{std}$$

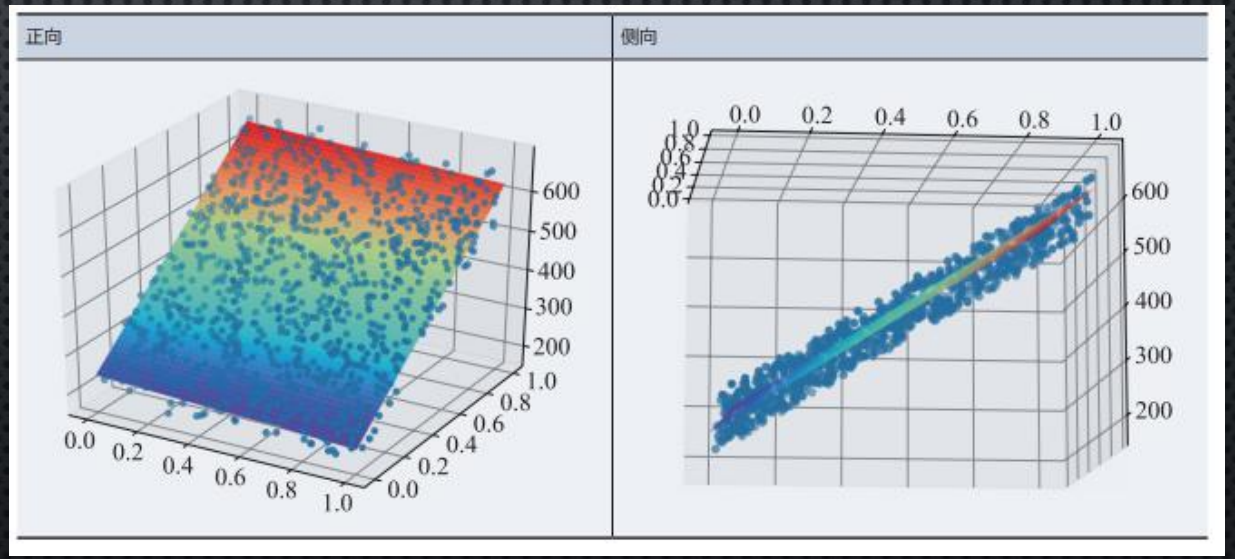
- 中心化：把每个特征值中的所有数据变成平均值为 0 的数据，无标准差要求。

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

# 5.4 样本特征数据标准化

- 代码运行结果
  - 损失函数图像：稳定在400左右震荡。
  - 神经网络预测房价结果为37366万元！但可视化结果表明，神经网络的训练并没有问题。



# 5.5 还原参数值

- 超参修改
  - 学习率缩小10倍。
  - Max_epoch 扩大50倍。
  - Batch_size 调整为10，小批量梯度下降提高精度，减缓个别样本引起的跳跃程度。
- 还原真实的$W,B$值

我们唯一修改的地方，就是样本数据特征值的标准化，并没有修改标签值。可以大概猜到W的值和样本特征值的缩放有关系，而且缩放倍数非常相似，甚至可以说一致。

# 5.5 还原参数值

➢ 数学推导

- 假设在标准化之前，真实的样本值是 $X$，真实的权重值是 $W$；在标准化之后，样本值变成了 $X'$，训练出来的权重值是 $W'$。标准化公式为：

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

- 预测值和标签值应该相等，即有：

$$y = x_1 w_1 + x_2 w_2 + b = x_1' w_1' + x_2' w_2' + b' = z$$

$$x_1' w_1' + x_2' w_2' + b' = \frac{x_1 - x_{min}}{x_{max} - x_{min}} w_1' + \frac{x_2 - x_{min}}{x_{max} - x_{min}} w_2' + b'$$

# 5.5 还原参数值

- 与原式相比，可以得到：

$$x_1 w_1 + x_2 w_2 + b = \frac{x_1 w_1'}{x_{max} - x_{min}} + \frac{x_2 w_2'}{x_{max} - x_{min}} + b' - \frac{x_{min}(w_1' + w_2')}{x_{max} - x_{min}}$$

$$w_1 = \frac{w_1'}{x_{max} - x_{min}}, \quad w_2 = \frac{w_2'}{x_{max} - x_{min}}, \quad b = b' - \frac{x_{min}(w_1' + w_2')}{x_{max} - x_{min}}$$

将实际数值代入计算，代码运行结果得到房价预测值 46.266 万元，与正规方程解相近。

如果遇到非线性问题，或者深层网络，这么做是不是也可以呢？

# 5.6 标签值标准化

- 预测数据的标准化
  - 我们只需要把训练数据的最小值和最大值记录下来，在预测时使用它们对预测数据做标准化，就相当于把预测数据“混入”训练数据。
  - 代码运行结果与正规方程解非常接近。我们只需把预测数据看作训练数据的一个记录，先做标准化，再做预测，这样就不需要把权重矩阵还原了。

看上去我们已经完美地解决了这个问题，但是仔细看看$loss$值，还有$w,b$的值，都是几十几百的数量级，这和神经网络的概率计算的优点并不吻合，实际上它们的值都应该在$[0,1]$之间的。

大数量级的数据有另外一个问题，就是它的波动有可能很大。目前我们还没有使用激活函数，一旦网络复杂了，开始使用激活函数时，像 486.166 这种数据，一旦经过激活函数就会发生梯度饱和的现象，输出值总为 1，这样对于后面的网络就没什么意义了，因为输入值都是 1。

# 5.6 标签值标准化

- 训练过程中的数值的数量级问题
  - 观察损失函数的结构：

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

损失函数值和反向传播链的数值都会很大。如果我们像对特征值做标准化一样，把标签值也标准化到[0,1]之间，是不是有帮助呢？

# 5.6 标签值标准化

- 标签值的标准化和反标准化
  - 对标签值进行标准化，有效降低了损失函数值和梯度的数量级，减少了迭代次数：

$$y_{new} = \frac{y - y_{min}}{y_{max} - y_{min}}$$

  - 得到预测结果后，需进行反标准化：

$$y = y_{new}(y_{max} - y_{min}) + y_{min}$$

# 5.6 标签值标准化

- 标准化总结
  - 样本不做标准化的话，网络发散，训练无法进行；
  - 训练样本标准化后，网络训练可以得到结果，但是预测结果有问题；
  - 还原参数值后，预测结果正确，但是此还原方法并不能普遍适用；
  - 标准化测试样本，而不需要还原参数值，可以保证普遍适用；
  - 标准化标签值，可以使得网络训练收敛快，但是在预测时需要把结果反标准化，以便得到真实值。

# THE END

谢谢！