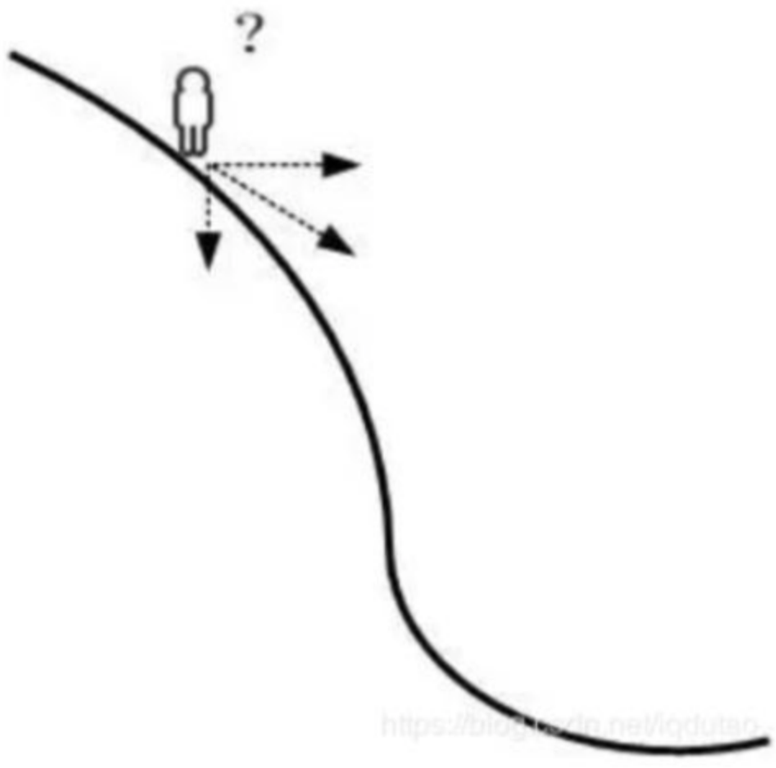
梯度就是导数，而梯度下降法就是一种通过求目标函数的导数来寻找目标函数最小化的方法。梯度下降目的是找到目标函数最小化时的取值所对应的自变量的值，目的是为了找自变量X。

最优化问题在机器学习中有非常重要的地位，很多机器学习算法最后都归结为求解最优化问题。最优化问题是求解函数极值的问题，包括极大值和极小值。在各种最优化算法中，梯度下降法是最简单、最常见的一种，在深度学习的训练中被广为使用。

## 1. 梯度下降理解

梯度下降法的基本思想可以类比为一个下山的过程。






按照梯度下降算法的思想，它将按如下操作达到最低点：

- 明确自己现在所处的位置
- 找到相对于该位置而言下降最快的方向
- 沿着第二步找到的方向走一小步，到达一个新的位置，此时的位置肯定比原来低
- 回到第一步
- 终止于最低点

按照以上5步，最终达到最低点，这就是梯度下降的完整流程。当然你可能会说，上图不是有不同的路径吗？是的，因为上图并不是标准的凸函数，往往不能找到最小值，只能找到局部极小值。所以可以用不同的初始位置进行梯度下降，来寻找更小的极小值点。

## 2. 算法解释

我们知道，对于一个逻辑回归函数，我们可以得到其代价函数，用代价函数来衡量模型预测值与真实值之间差异的函数。

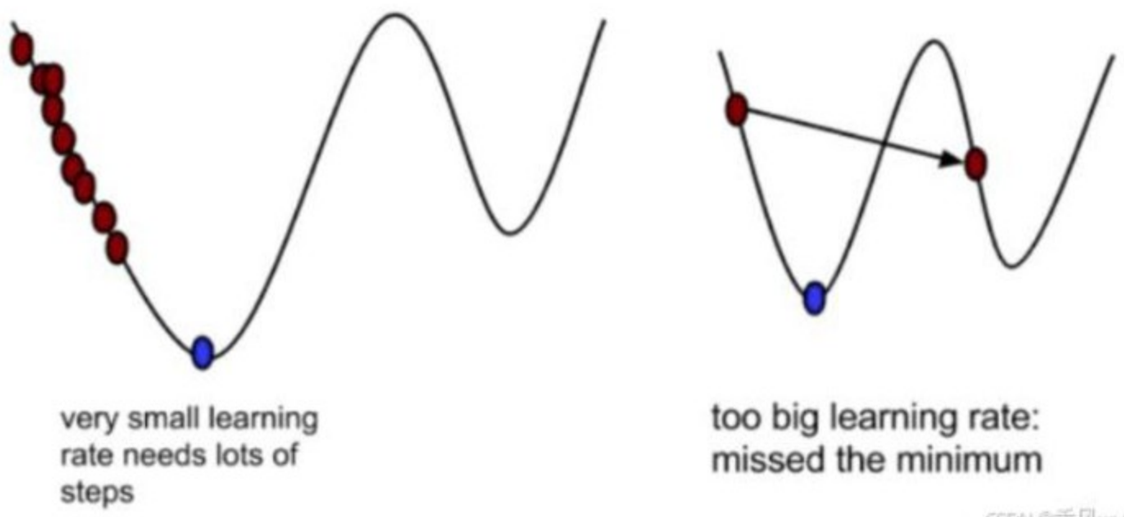
$$J(w,b)=\frac{1}{m}\sum_{i=1}^{m}L(\widehat{y}^{(i)},y^{(i)})=-\frac{1}{m}\sum_{i=1}^{m}y^{(i)}log\widehat{y}^{(i)}+(1-y^{(i)})log(1-\widehat{y}^{(i)})$$

定义一个公式如下，J是关于w和b的一个函数，我们在山林里当前所处的位置为（$w_0,b_0$）点，要从这个点走到J的最小值点，也就是山底。首先我们先确定前进的方向，也就是梯度的反向，然后走一段距离的步长，也就是α，走完这个段步长，就到达了（$w_1,b_1$）这个点。

$$w:=w-\alpha\frac{dJ(w,b)}{dw}$$

$$b:=b-\alpha\frac{dJ(w,b)}{db}$$

α在梯度下降算法中被称作为学习率（learning rate）或者步长（stride），意味着我们可以通过α来控制每一步走的距离，以保证不要步子跨的太大，其实就是不要走太快，错过了最低点。同时也要保证不要走的太慢，导致太阳下山了，还没有走到山下。所以α的选择在梯度下降法中往往是很重要的，α不能太大也不能太小，太小的话，可能导致迟迟走不到最低点，太大的话，会导致错过最低点。

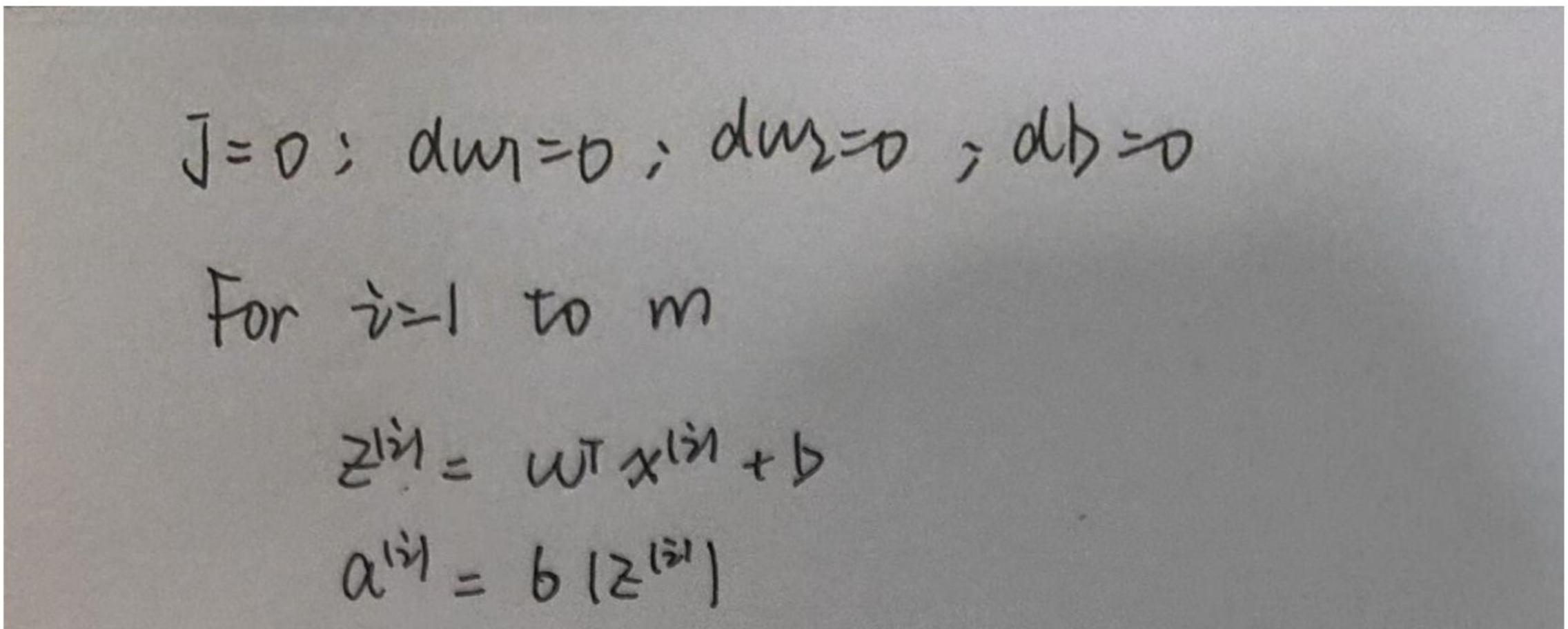



## 3. m个样本的梯度下降

损失函数$J(w,b)$的定义如下：

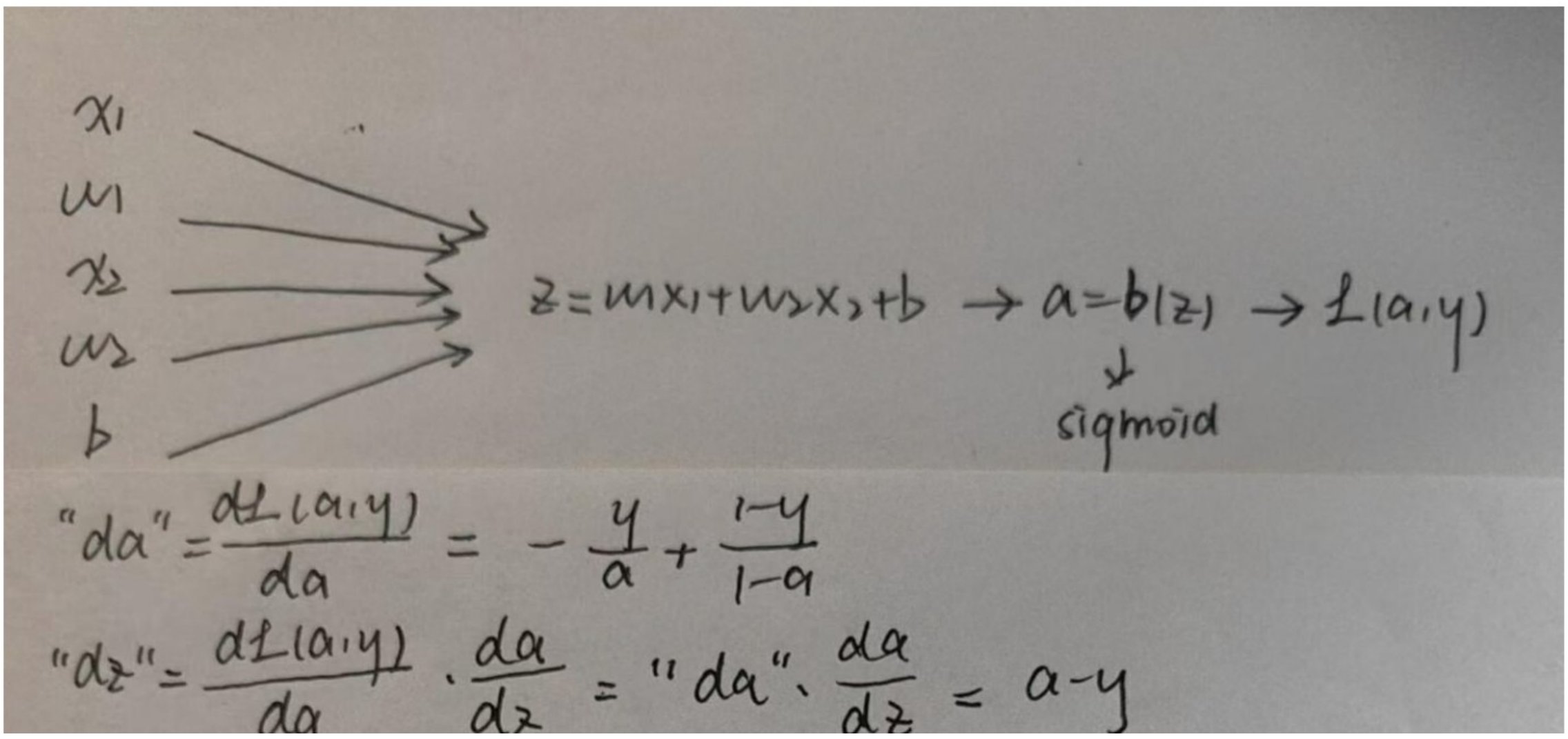
$$\mathrm{J(w,b)}=\frac{1}{\mathrm{m}}\sum_{\mathrm{i=1}}^{\mathrm{m}}\mathrm{L(a^{(i)},y^{(i)})}$$

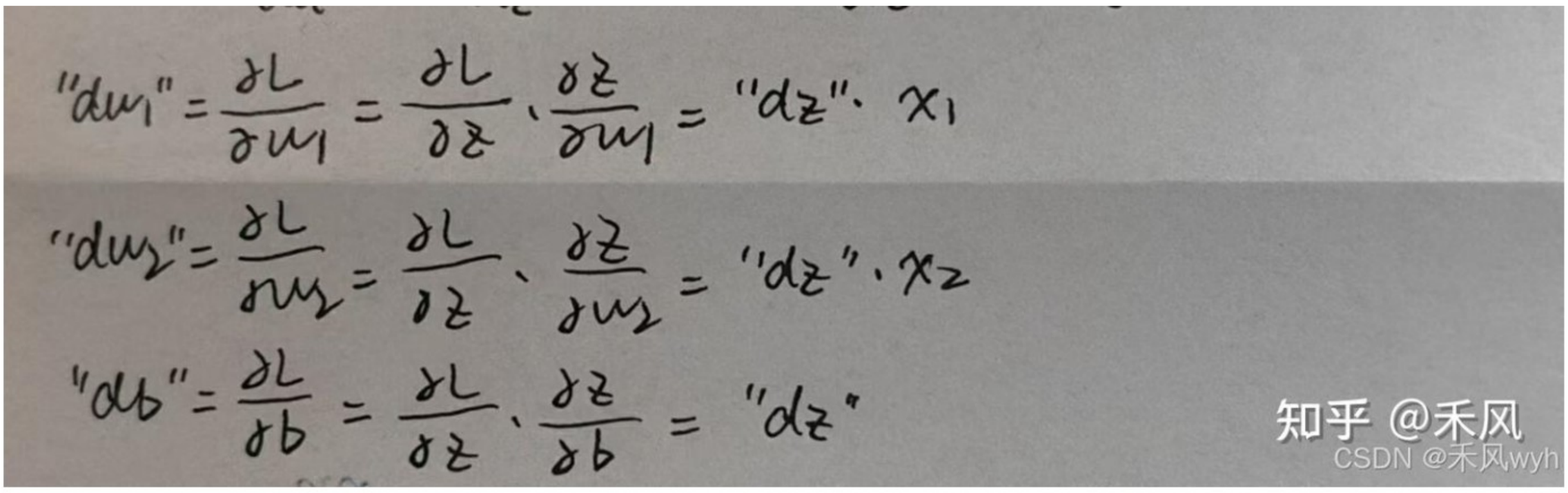
当算法输出关于样本y 的 $a^{(i)}$，$a^{(i)}$是训练样本的预测值，即：$\sigma(z^{(i)})=\sigma(w^Tx^{(i)}+b)$。在前面展示的是对于任意单个训练样本 $(x^{(i)},y^{(i)})$，$dw_1$，$dw_2$ 和 $db$ 添上上标 i 表示你求得的相应的值。带有求和的全局代价函数，实际上是1到m 项各个损失的平均。 所以它表明全局代价函数对 $w_1$ 的微分，对 $w_1$ 的微分也同样是各项损失对 $w_1$ 微分的平均。




$$J/ = m, dw_1/ = m, dw_2/ = m, db/ = m$$

为什么 $dz$ 、 $dw_1$ 、 $dw_2$ 、 $db$ 表达式是这样的呢?






## 4. 代码

```
J=0;dw1=0;dw2=0;db=0;
for i = 1 to m
    z(i) = wx(i)+b;
    a(i) = sigmoid(z(i));
    J += -[y(i)log(a(i))+(1-y(i)) log(1-a(i));
    dz(i) = a(i)-y(i);
    dw1 += x1(i)dz(i);
    dw2 += x2(i)dz(i);
    db += dz(i);
J/= m;
dw1/= m;
dw2/= m;
db/= m;
w=w-alpha*dw
b=b-alpha*db
```