

总结

到了现在，大家通过简单的例子以及穿插的代码对神经网络的搭建和训练有了最初步的认识。但是之前的内容仅仅是为了让大家略知一二。当大家对整体过程有了一定认识之后，我们再对每一个地方进行细节的讲解。否则，如果像传统的书籍一样，按顺序对每一个点进行深入而且严谨的讲解，会让读者不清楚这样的目的是什么。

扩展

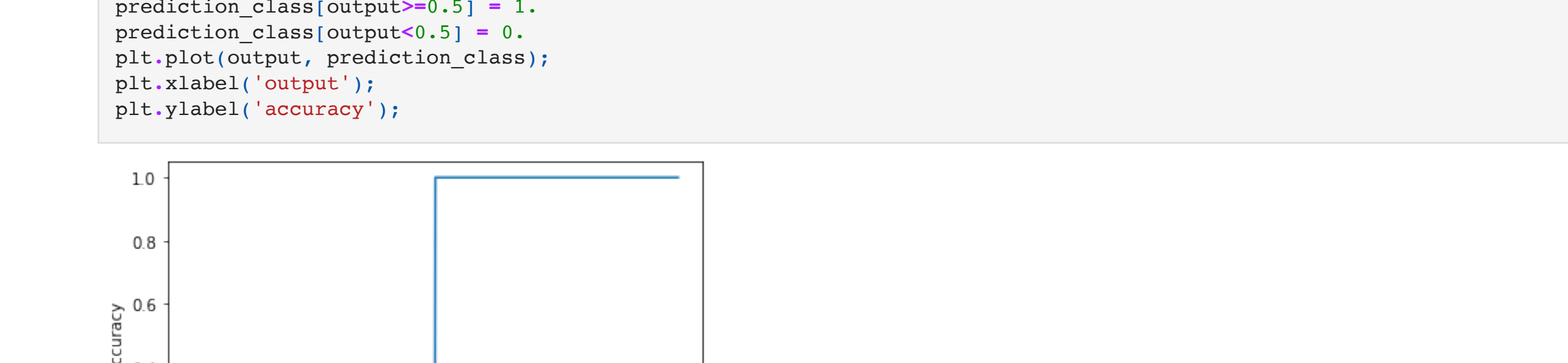
现在，我们基于之前学习的关于机器学习的基本知识和搭建的初步神经网络对内容进行扩展。让算法和代码的每一部分都变得越来越严谨。要注意的是，这里的内容仍然只是整个机器学习的冰山一角。

损失函数

损失函数又叫目标函数。所以显而易见，损失函数应当直接表现任务的目标。下面我们来看一下分类和回归任务中的损失函数：

分类

在一般的分类任务中，我们的最直观目标就是增加分类的准确率。但是很遗憾，准确率是一个不可导的标准。让我们以二分类问题为例子，我们把归一化之后的输出小于0.5的情况分类为0，大于0.5的情况分类为1，那么对于每一个数据，准确率和输出之间的关系就是：



可见accuracy对于输出output的梯度处处为0，那么梯度无法反向传播。因此我们需要一个连续的函数用于代替准确率来作为损失函数。这时候，交叉熵被提了出来。我们先来看一个简单的函数：

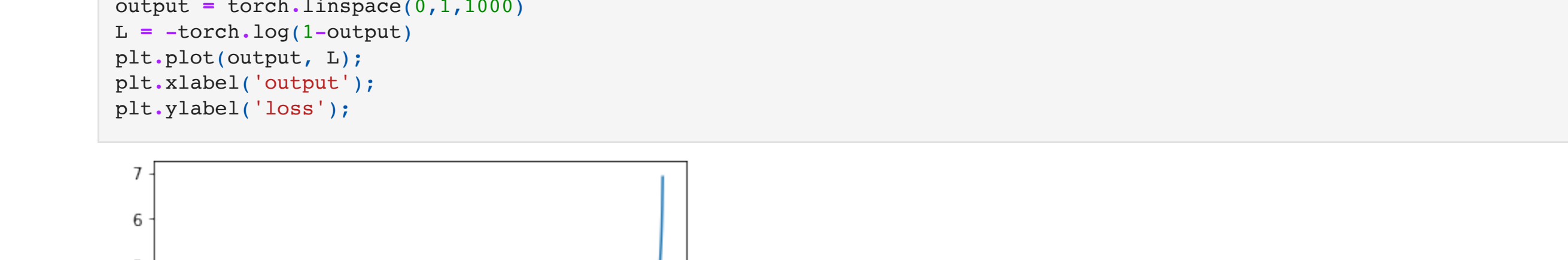
$$\mathcal{L} = -\log(\hat{y})$$



这样的话，当我们最小化损失函数时，就会把输出的值增加，这应该对应于分类为1的情况。对于分类应该为0的情况，我们其实应该减少输出值，这时候，我们让函数为

$$\mathcal{L} = -\log(1 - \hat{y})$$

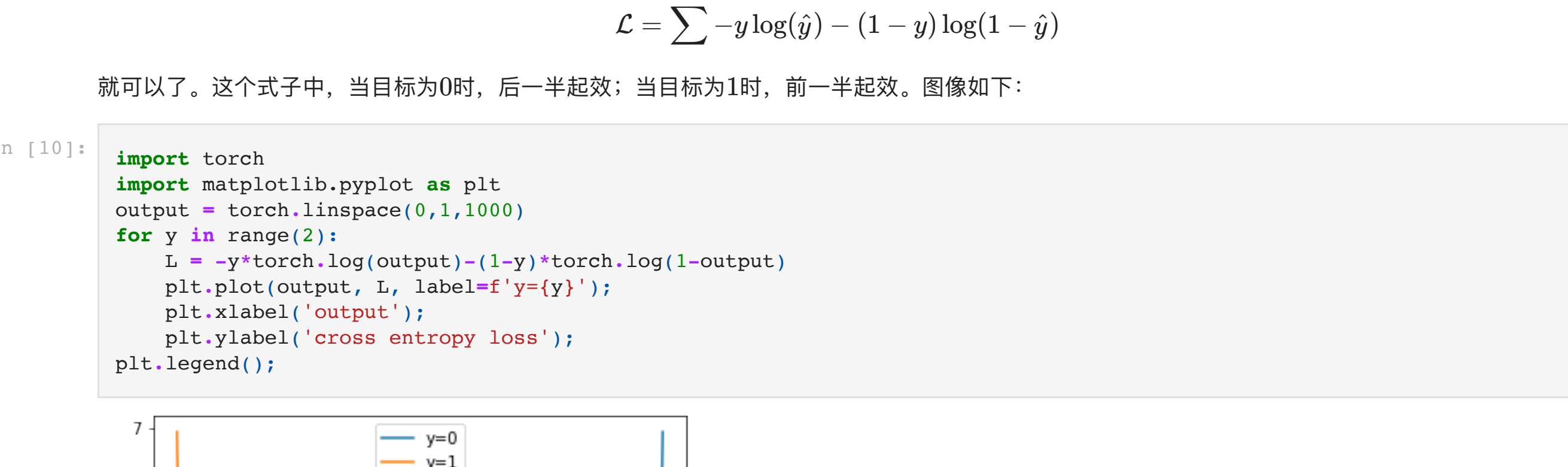
这样，损失函数就是



这样我们就有了对于0和1类的损失函数。这其实也可以用于多类的分类问题：经过one-hot编码之后，错误的分类的目标值为0，正确分类的目标值为1，所以我们只需要让

$$\mathcal{L} = \sum -y\log(\hat{y}) - (1 - y)\log(1 - \hat{y})$$

就可以了。这个式子中，当目标为0时，后一半起效；当目标为1时，前一半起效。图像如下：



这就是交叉熵函数(cross entropy loss)。

回归

回归 (regression) 问题又叫拟合 (aproximation) 问题，指的是根据给定的输入，让机器学习模型给出期望的函数值。对于这种问题，我们往往用输出值和目标值的差作为损失函数。例如

$$\mathcal{L} = (\hat{y} - y)^2 = \Delta^2$$

或

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

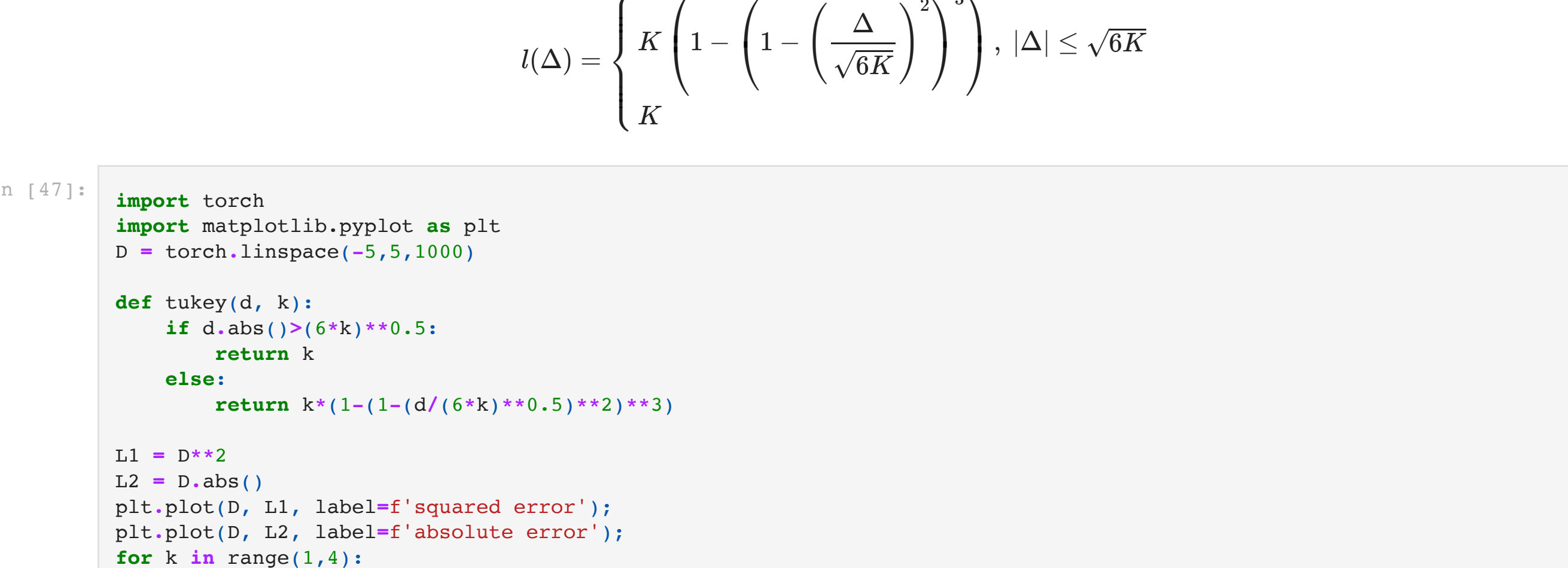
我们可以可视化一下这两个函数



可以看出，当我们把loss降低时，两个函数都会把Δ推向0。区别是：用平方作为损失函数时，当误差较小时，梯度较小。所以在学习过程中，当参数需要在多个训练数据间进行权衡的时候，误差大的会被着重考虑，也就是大误差的数据会对梯度有更大的影响。而绝对值作为损失函数时，所有的误差，无论本身的大小，对于梯度的影响都一样，也就是说，他们对于学习过程的影响是一样的。所以，相比之下，前者的训练结果更倾向于消除大的误差，而后者相比于前者，更倾向于把小的误差消除到0。

但是，我们在采集数据时，往往会出现特殊的情况，例如当我们采集身高的时候，出现了一个身高为58米的人，显然这是一个错误的数据，如果我们强行把这个数据考虑到数据集中，是愚蠢的。这种数据我们往往叫他们outlier。有的时候，由于问题的复杂，我们很难直观挑出他们，因此我们会想办法在训练过程中忽略那些误差特别离谱的数据点。这时候我们就要用 tukey loss

$$l(\Delta) = \begin{cases} K \left(1 - \left(1 - \left(\frac{\Delta}{\sqrt{6K}} \right)^2 \right)^3 \right), & |\Delta| \leq \sqrt{6K} \\ K & \end{cases}$$



激活函数

常见激活函数

非线性性

凹凸性

可学习激活函数

参数初始化

数值优化的初始值对数值优化的结果有巨大影响。而这个问题，我们也可以从不同的角度来看待。

随机初始化

Xavier

对于深度神经网络，会存在这样一个问题，就是当层数过多时，输出的值以及分布会越来越不正常。例如在加权求和和中 $Z = XW$ ，如果 W 的数学期望大于1，那么 Z 就会比 X 大一点，那么每一层的输出值都会增加一些，而且值的分布方差也会增加一些，这样经过多层之后会导致许多问题，例如梯度爆炸或者梯度消失。Xavier初始化的目标就是通过选择合适的权重 W ，使得每一层输出和输入的分布一致，这样的话无论传递多少层，都不会出现极端的情况。

要注意的是，Xavier初始化假设的是激活函数经过零点，且在零点处斜率为1。同时，假设输入 X 和权重 W 都是期望为0，方差很小的变量，因此 Z 也是期望为0，方差很小的变量。这样的话，一个期望值为0，方差较小的随机变量经过激活函数后，分布不变。

我们把输出 Z 表示为 $\mathbb{E}\{Z\} + \delta_Z$ ，其中 $\mathbb{E}\{Z\} = 0$ ，并且 δ_Z 是一个期望为0方差为 $\text{Var}(Z)$ 的随机变量。那么，激活函数的输出是：

$$\text{act}(\mathbb{E}\{Z\} + \delta_z) \approx \text{act}(\mathbb{E}\{Z\}) + \frac{\partial \text{act}(\mathbb{E}\{Z\})}{\partial Z} \delta_z = \delta_z$$

所以经过激活函数之后的变量的期望和方差是：

$$\begin{aligned} \text{Var}(\text{act}(\mathbb{E}\{Z\} + \delta_z)) &= \text{Var}(\delta_z) = \text{Var}(Z) \\ \mathbb{E}\{\text{act}(\mathbb{E}\{Z\} \pm \delta_z)\} &= \mathbb{E}\{\mathbb{E}\{Z\} + \delta_z\} = 0 = \mathbb{E}\{Z\} \end{aligned}$$

在这种假设之后，我们就可以忽略激活函数对于分布的影响了。这种情况下，我们假设输入是

$$X \in \mathbb{R}^{M \times E}$$

权重矩阵是

$$W \in \mathbb{R}^{N \times M}$$

输出是

$$Z = WX \in \mathbb{R}^{N \times E}$$

它的方差就是

$$\text{Var}(Z) = \frac{1}{E}(WX - \bar{W}\bar{X})(WX - \bar{W}\bar{X})^\top$$

根据假设， W 和 X 的数学期望是0，所以

$$\begin{aligned} \text{Var}(Z) &= \frac{1}{E}(WX)(WX)^\top \\ &= \frac{1}{E}WXX^\top W^\top \\ &= \frac{1}{E}W \cdot E \cdot \text{Var}(X) \cdot W^\top \end{aligned}$$

由于各个特征之间相互独立，并且已经假设每个特征的方差都是一样，记为 σ^2 ，我们简化它为 $\text{Var}(Z) = \sigma^2 I$ 和 $\text{Var}(X) = \sigma^2 I$ 。其中 I 是单位矩阵，维度根据所乘对象而改变。所以：

$$\begin{aligned} \sigma^2 I &= W \cdot \sigma^2 I \cdot W^\top \\ I &= WW^\top \\ &= M \cdot \text{Var}(W) \\ \text{Var}(W) &= \frac{I}{M} \end{aligned}$$

其中 M 是输入特征个数。

在工程上也有用输入和输出的平均值，也就是 $\text{Var}(W) = \frac{2I}{M+N}$ 作为权重初始方差的，这没有什么数学原理，只是工程经验。

随机种子对初始化的影响

优化策略和优化器

分批训练

SGD

动量

自适应长度

学习率

固定学习率

可变学习率

数据集

预处理

数据集分割

Early stop

正则化

实验的可重复性

方便自己

方便他人

