Huber LOss

Huber损失结合了MSE和MAE的最佳特性。对于较小的误差，它是二次的，否则是线性的(对于其梯度也是如此)。Huber损失需要确定δ参数：

$$L_\delta = \begin{cases} \frac{1}{2}(y - f(x))^2, \text{ if } |y - f(x)| \le \delta \\ \delta|y - f(x)| - \frac{1}{2}\delta^2, \quad \text{otherwise} \end{cases}$$

代码：

```
def update_weights_Huber(m, b, X, Y, delta, learning_rate):
    m_deriv = 0
    b_deriv = 0
    N = len(X)
    for i in range(N):
        # 小值的二次导数，大值的线性导数
        if abs(Y[i] - m*X[i] - b) <= delta:
            m_deriv += -X[i] * (Y[i] - (m*X[i] + b))
            b_deriv += - (Y[i] - (m*X[i] + b))
        else:
            m_deriv += delta * X[i] * ((m*X[i] + b) - Y[i]) / abs((m*X[i] + b) - Y[i])
            b_deriv += delta * ((m*X[i] + b) - Y[i]) / abs((m*X[i] + b) - Y[i])
    #我们减去它，因为导数指向最陡的上升方向
    m -= (m_deriv / float(N)) * learning_rate
    b -= (b_deriv / float(N)) * learning_rate
    return m, b
```

Huber损失对于异常值比MSE更强。**它用于稳健回归(robust regression)，M估计法(M-estimator)和可加模型(additive model)。Huber损失的变体也可以用于分类。**

MAE Loss

绝对损失误差，每个训练样本的绝对误差是预测值和实际值之间的距离，与符号无关。绝对误差也称为**L1 Loss**：

$$L = |y - f(x)|$$

与MSE相比，MAE成本对异常值更加健壮。但是，在数学方程中处理绝对或模数运算符并不容易。我们可以认为这是MAE的缺点。以下是MAE成本更新权重的代码

```
def update_weights_MAE(m, b, X, Y, learning_rate):
    m_deriv = 0
    b_deriv = 0
    N = len(X)
    for i in range(N):
        #计算偏导数
        # -x(y - (mx + b)) / |mx + b|
        m_deriv += - X[i] * (Y[i] - (m*X[i] + b)) / abs(Y[i] - (m*X[i] + b))
        # -(y - (mx + b)) / |mx + b|
        b_deriv += -(Y[i] - (m*X[i] + b)) / abs(Y[i] - (m*X[i] + b))
```

```
    #我们减去它，因为导数指向最陡的上升方向
    m -= (m_deriv / float(N)) * learning_rate
    b -= (b_deriv / float(N)) * learning_rate
    return m, b
```

Hinge Loss

在机器学习中，**Hinge loss**（铰链损失函数）作为损失函数，通常被用于最大间隔算法(maximum-margin)，而最大间隔算法又是SVM(支持向量机support vector machines)用到的重要算法。

Hinge loss专用于二分类问题，标签值y=±1，预测值$\hat{y}$∈R。该二分类问题的目标函数的要求如下：当$\hat{y}$等于+1或者小于等于-1时，都是分类器确定的分类结果，此时的损失函数loss为0；而当$\hat{y}$预测值∈(−1,1)时，分类器对分类结果不确定，loss不为0。显然，当y^=0时，loss达到最大值。

python代码实现

```
def update_weights_Hinge(m1, m2, b, X1, X2, Y, learning_rate):

    m1_deriv = 0
    m2_deriv = 0
    b_deriv = 0
    N = len(X1)
    for i in range(N):
        # 计算偏导数
        if Y[i]*(m1*X1[i] + m2*X2[i] + b) <= 1:
            m1_deriv += -X1[i] * Y[i]
            m2_deriv += -X2[i] * Y[i]
            b_deriv += -Y[i]
        # 否则偏导数为0
    # 我们减去它，因为导数指向最陡的上升方向
    m1 -= (m1_deriv / float(N)) * learning_rate
    m2 -= (m2_deriv / float(N)) * learning_rate
    b -= (b_deriv / float(N)) * learning_rate
return m1, m2, b
```

池化方法补充

随机池化(Stochastic Pooling)

Stochastic pooling是**一种简单有效的正则化CNN的方法，能够降低max pooling的过拟合现象，提高泛化能力**。对于pooling层的输入，根据输入的多项式分布随机选择一个值作为输出。

训练阶段：

1）前向传播：先将池化窗口中的元素全部除以它们的和，得到概率矩阵；再按照概率随机选中的方格的值，作为该区域池化后的值。

2）反向传播：求导时，只需保留前向传播中已经被选中节点的位置的值，其它值都为0，类似max-pooling的反向传播。

测试阶段：

在测试时也使用Stochastic Pooling会对预测值引入噪音，降低性能。取而代之的是使用概率矩阵加权平均。比使用Average Pooling表现要好一些。在平均意义上，与Average Pooling近似，在局部意义上，服从Max Pooling准则。

数据增强方法补充

1.几何变换类

几何变换类即对图像进行几何变换，包括**翻转，旋转，裁剪，变形，缩放**等各类操作。

2.颜色变换

包括**噪声、模糊、颜色变换、擦除、填充**等等。

3.GAN

通过生成对抗网络生成同类型的数据。比如生成汽车、人脸图片。通过图像风格迁移的手段，还可以生成同一物体再不同环境下的图片。

图像分类方法综述

传统方法

传统方法通常完整建立图像识别模型一般包括底层特征学习、特征编码、空间约束、分类器设计、模型融合等几个阶段。

1). 底层特征提取: 通常从图像中按照固定步长、尺度提取大量局部特征描述。常用的局部特征包括SIFT(Scale-Invariant Feature Transform, 尺度不变特征转换) 、HOG(Histogram of Oriented Gradient, 方向梯度直方图) 、LBP(Local Bianray Pattern, 局部二值模式)等，一般也采用多种特征描述，防止丢失过多的有用信息。

2). 特征编码: 底层特征中包含了大量冗余与噪声，为了提高特征表达的鲁棒性，需要使用一种特征变换算法对底层特征进行编码，称作特征编码。常用的特征编码方法包括向量量化编码、稀疏编码、局部线性约束编码、Fisher向量编码等。

3). 空间特征约束: 特征编码之后一般会经过空间特征约束，也称作特征汇聚。特征汇聚是指在一个空间范围内，对每一维特征取最大值或者平均值，可以获得一定特征不变形的特征表达。金字塔特征匹配是一种常用的特征汇聚方法，这种方法提出将图像均匀分块，在分块内做特征汇聚。

4). 通过分类器分类: 经过前面步骤之后一张图像可以用一个固定维度的向量进行描述，接下来就是经过分类器对图像进行分类。通常使用的分类器包括SVM(Support Vector Machine, 支持向量机)、随机森林等。而使用核方法的SVM是最为广泛的分类器，在传统图像分类任务上性能很好。

深度学习方法

Alex Krizhevsky在2012年ILSVRC提出的CNN模型取得了历史性的突破，效果大幅度超越传统方法，获得了ILSVRC2012冠军，该模型被称作AlexNet。这也是首次将深度学习用于大规模图像分类中。从AlexNet之后，涌现了一系列CNN模型，不断地在ImageNet上刷新成绩，如图5展示。随着模型变得越来越深以及精妙的结构设计，Top-5的错误率也越来越低，降到了3.5%附近。而在同样的ImageNet数据集上，人眼的辨识错误率大概在5.1%，也就是目前的深度学习模型的识别能力已经超过了人眼。

深度学习的图像分类算法主要基于卷积神经网络CNN。典型的CNN分类网络有AlexNet、VGG、GoogleNet、ResNet等。