CS231n学习#1：KNN与线性分类

原创 Zerg_Wang

1.KNN

PS：文中所述的“label”、“标签”、“类别”是指同一个意思。

本文所述的图像识别，是指通过训练机器，使其可以判断出给定照片呈现的是什么内容。实现此功能一般需要两个步骤：

```
def train(images,labels)
    '''
    训练出可以对图像进行分类的模型
    '''
    return model


def predict(model,test_images)
    '''
    利用训练出的模型完成预测任务
    '''
    return test_labels
```

而在训练过程中需要用到一些比较函数，来判断两图相似情况，这里列举几个判断方法：

1.曼哈顿距离（L1距离）：图片1和图片2每个像素值的差的绝对值之和。

$$L_1 = \sum(|x_i - y_i|)$$

2.欧式距离（L2距离）：图片1和图片2所有像素值的差的平方和再开方。

$$L_2 = \sqrt{\sum(x_i - y_i)^2}$$

3.切比雪夫距离（L∞距离）：图片1和图片2相差最大的两个像素的差值。

$$L_\infty = \sqrt[\infty]{\sum(x_i - y_i)^\infty} = max(x_i - y_i)$$

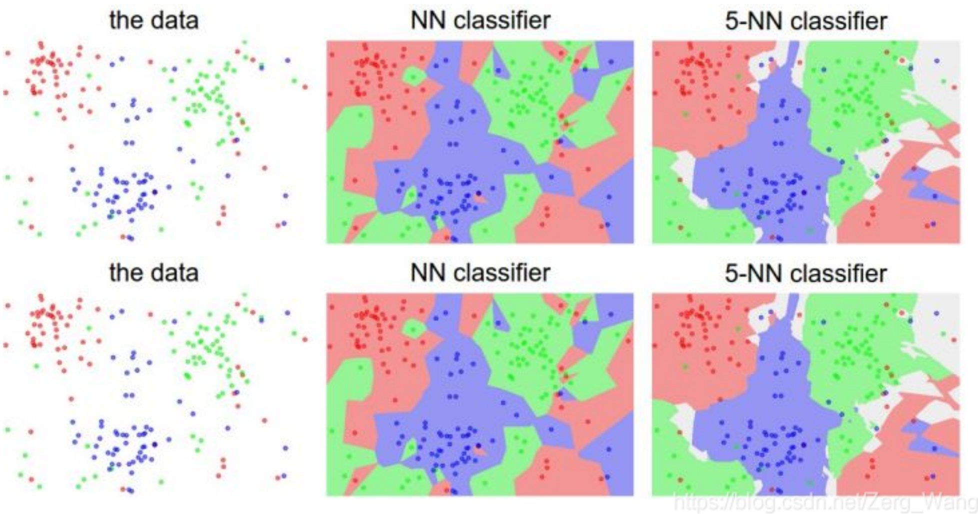
4.闵科夫斯基距离（Minkowski Distance）：由上式推广得到。

$$L_k = \sqrt[k]{\sum(x_i - y_i)^k}$$

通过以上比较函数，可使用KNN（K-Nearnest Neighbour）完成最简单的图像分类任务。

对于给出的需要预测的图片，将该图与数据集中的所有图片进行比较，找出与之最相近的k张，然后根据这k张图片的label，确定预测图片的label。

例如，对于图片A，设k=6，与之最相近的六张图片B、C、D、E、F、G的label分别是猫、猫、狗、牛、狗、猫，则猫有3票，狗有2票，牛有一票，则图A的预测结果为猫。（实际上，KNN不一定是以这k张图片的label来确定，也有可能对k张照片赋以权重或取均值来决定预测图片的类别）

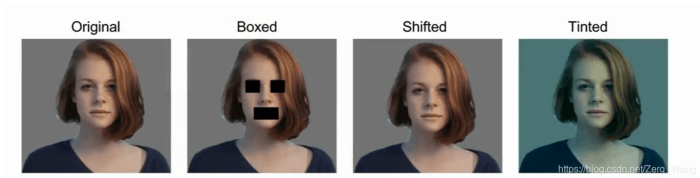


以上图为例，左边是数据集，每个点代表一张图，点的颜色表示其类别，中间是k=1时的KNN，这张五颜六色的图是如何得出的呢，对于左图上任意空白位置，假设在此处投下一个点（代表一张图），然后根据KNN判断其颜色（根据最临近的k个有颜色的点判断），以此得到该空白处的颜色。

对比中间和右边的图可知，当k越大，分类的边界也就越平滑，说明算法越稳定，泛化能力更强，不易受到特殊值影响。然而，右图中也出现了空白，说明此处的点无法被归类。

因此，对于不同的k，模型预测能力也将不同，为得到最好的模型，需要确定k的值，这个k就是该模型的超参数。（超参数是在学习前需要人工确定的参数，最优的超参数可能需要通过枚举等方式来确定）

KNN很少应用于实际，一方面是因为其不需要事先train，所有的运算都在predict环节，这样一来，假设使用KNN算法进行大量的预测任务，其效率无疑是极低的。另一方面是其准确率过低。需要分类的图像一般较为复杂，图片间的相似程度无论是用L1还是L2距离来计算都是极为不合适的。



例如上图，通过人工识别会认为上面的四张图是相似的，但同过L1、L2对比函数会得到相当大的数值，以此得到的结果运用于KNN中将会导致不正确的结果。

2.线性分类

不同于KNN，为了将运算集中在train，而减少predict环节的运算，可采用线性分类器（Linear Classifier）来解决图像分类问题。该分类器由两部分组成：

评分函数（score function）

输入图片，输出其经预测得到的标签。实际上，若某个数据集中有多种可能的标签（假设有n个），那么输出的会是n个值，表示第i个标签的可能性，值越高表示图片属于该类的可能性越大。

以最简单的评分函数——线性分类器为例：

$$f(x_i, W, b) = Wx_i + b$$

该函数中的运算均为矩阵运算，Xi为输入的图片，假设该图片是32×32的RGB图像，那么它会被转换为一个3072×1的向量（3072=32×32×3）。而W为权重向量，假设图片可能的label有10个，那么W为一个10×3072的矩阵。b为偏差向量（bias vector），它影响输出数值，但是并不和xi产生关联。（实际上，针对某一label，可单独训练一个线性分类器，但这10个分类器也可以通过增加W的维度进行合并，因此大小为10×3072的W中每一个行向量就是某个单一线性分类器的W）

更进一步，可以将b当作W的一个列向量从而合并到W中，这样一来，W大小为10×3073，但是输入的图像也需要进行处理：3073×1，多出的一维中存储常量1，从而不影响原有的运算。就此，评分函数可简化为：

$$f(x_i, W) = Wx_i$$

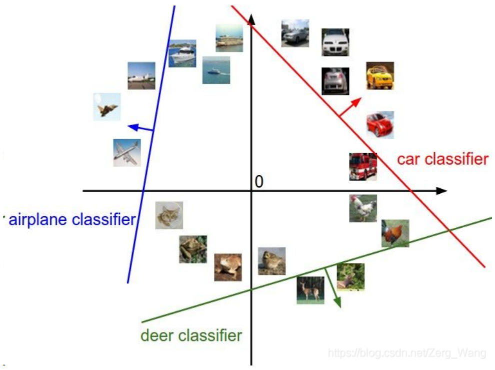
为什么仅凭一个权重矩阵就能得出图片属于某一类别的可能性？以下是关于线性分类器的理解：

1.W的参数

通过最优化过程，W的参数会不断调整，这些参数有正有负，会影响图片的分类情况，例如W中的某个行向量是用来确定图片与“船”这一类别的关系，那么其参数中有关蓝色这一项的数值会很高（即RGB中“B”这一项的参数值会较高，而“R”和“G”会相对偏低，甚至是负数）。因此通过W的参数可在一定程度上解释模型。

2.高维点与线性分类

在线性分类器中我们就图片处理为3073×1的向量，其实可以把该图片视为3073维空间中的一个点，这样一来，线性分类器就相当于该空间中的一条分割线：



图中有3个线性分类器，即W由3个行向量构成，每个Wi决定了线的斜率，每个b的改变会导致分割线上下平移，因此最优化过程也可以看做是找到最适合的参数，使图片（即上图中的点）以最优的方式按其实际类别分隔开。

3.权重视为模板

W的每一个行向量其实可以看作对应的类别的模板（有时候称为原型），下图为本人在cs231n的assignment 1的作业中根据CIFAR-10数据集训练出的W：

与上文所述一致，“ship”这一类的“模板”的确是蓝色偏多，而“car”看起来像一辆红车的车头。在W与xi进行点积的时候，其实就是在计算图片与哪个模板最为接近。

目标函数（Object Function）

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + \lambda R(W)$$

先提一下在神经网络中以下名词的定义：

损失函数（Loss Function）：即Li，表示单个样本（图片）的预测值与实际值的偏差。

代价函数（Cost Function）：整个数据集预测值与实际的偏差，即：

$$\frac{1}{N}\sum_{i=1}^{N} L_i$$

目标函数（Object Function）：即L，它等于经验风险+结构风险，其中，经验风险即代价函数，结构风险即正则化项。

然后我们来看以上公式：

N表示数据集中共有N张图片，Li表示第i张图片经评分函数得到的标签与真实标签之间的差异值，值越大，表示评分函数准确度越低。该方法可转化为一个最优化问题，在最优化过程中，将通过更新评分函数的参数来最小化损失函数值（也就是更新W和b中的参数，更新的过程其实就是

train，一旦train环节完成，原始数据集就可以丢弃，进行predict的应用）。

R(W)为正则化，损失函数在最优化过程中可能会发现有多种W可以正确分类图像，而正则化有使得损失函数倾向于选择更为简单的W。

L1正则化将W的每个值的绝对值相加，在最优化过程中会使得W整体趋向于0：

$$R_{L1}(W)=\sum_l\sum_r|W_{l,r}|$$

L2正则化最为常用，它将W的每个值的平方相加，在最优化过程中会使得W整体趋向于0且使得数值较为平均（即方差较小）。

$$R_{L2}(W)=\sum_l\sum_r W_{l,r}^2$$

L2正则化使得W的参数方差较小，可提高其泛化能力。

例如，假设输入的图片$X=[1,1,1,1]$, $W_1=[1,0,0,0]$, $W_2=[0.25,0.25,0.25,0.25]$

虽然W1和W2与X相乘的结果都为1，但明显W2经L2正则化后的损失函数比W1低得多，因为W2的参数更小更分散，利于分类器将更多维度的信息利用起来，而不是仅仅使用少数几个维度的信息，从而提高分类器泛化能力。

以下介绍两种评估W好坏的损失函数（即Li）：

1.SVM（Support Vector Machine Loss）

支持向量机损失函数，其公式为：

$$L_i=\sum_{j\neq r}^{k}max(0,s_j-s_r+t)$$

其中，Li为数据集中第i张照片的损失值，k表示可能的label有k个，Sj表示第i张图片经过评分函数后在第j个label下的得分，r表示第i张图片实际的label是第r个，t为安全阈值（这个为超参数）。综上，该函数的思路就是：对于该图片所有不正确标签下的得分，若该得分比正确项的得分高且高出了安全阈值t，那就会将超出部分加到Li中，因此来“惩罚”不合格的W。

实际上，sj其实就是W的第j个行向量与图片运算的结果，因此上式也可以写成：（Xi表示第i张图片）

$$L_i=\sum_{j\neq r}^{k}max(0,W_jX_i-W_rX_i+t)$$

2.Softmax

$$L_i=\frac{e^{s_r}}{\sum_{j=1}^{k}e^{s_j}}$$

Sj、Sr的意义同SVM，此外Li还表示图片i被正确分类（被分类为r）的概率，其值域为(0,1)。

在计算Li的时候，因为涉及到指数，运算过程中可能会出现较大的数字，因此一般会在分子分母上先同除以$e^{max(s_1,s_2,s_3\dots)}$再运算。

除了以上形式的Softmax损失函数，还有对数形式的：

$$L_i=-ln(\frac{e^{s_r}}{\sum_{j=1}^{k}e^{s_j}})=-s_r+ln(\sum_{j=1}^{k}e^{s_j})$$

PS：对于带对数的Softmax，我一直不确定其底数到底是多少，网上的答案不尽相同，在此先采用底数为e的形式。

因为概率的范围为(0,1)，且Li单调递减，因此被正确分类的概率越高，Li的值越小，Li的值域为(0,+∞)，这样一来，对于分类错误的惩罚就不再局限于(0,1)之内了，错得越离谱，惩罚越重（惩罚没有上限！），因此这种形式的损失函数可能会有更好的表现。

Softmax与SVM的异同：一般来说这两种损失函数差别不大，然而SVM更看重“整体”，对数据的细节并不关系，举个例子：

假如说某图片经过W1、W2的输出分别为：10，9，9和10，2，2。假设正确分类是10，那无论是W1还是W2，SVM得到的损失函数值都是0，然而对于Softmax，W1的Loss要比W2高出不少。对于SVM来说，只要其结果在可接受范围内即可，没有更高的要求，然而对于Softmax来说，仅仅

是“达标”还不行，还需要“精益求精”，因此对于一些较为细致的图像分类任务可能会有更好的表现。

本篇博文知识基于斯坦福大学CS231n课程，经博主学习、整理而来，仅为个人理解，若有错误，欢迎批评指正，谢谢！

部分图片及思路来源：

http://cs231n.stanford.edu/

https://zhuanlan.zhihu.com/p/21930884

附上cs231n的网易云课堂地址：

https://study.163.com/course/courseMain.htm?courseId=1004697005