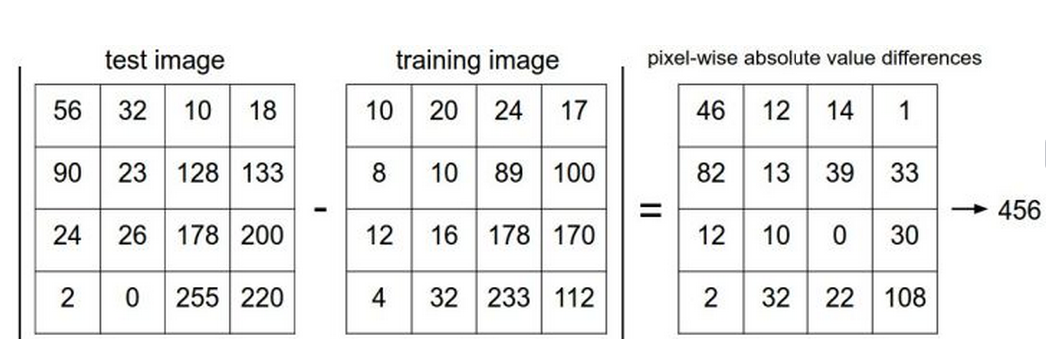
图像分类

Nearest Neighbor分类器：



Nearest Neighbor算法将会拿着测试图片和训练集中每一张图片去比较，然后将它认为最相似的那个训练集图片的标签赋给这张测试图片。方法是逐个像素求差值，所有差值相加

**for** i **in** xrange(num\_test):

distances **=** np**.**sum(np**.**abs(self**.**Xtr **-** X[i,:]), axis **=** 1)

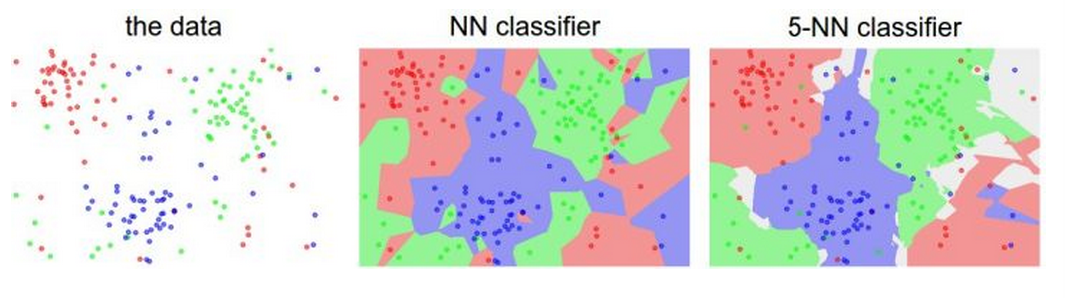
min\_index **=** np**.**argmin(distances) *# get the index with smallest distance*

Ypred[i] **=** self**.**ytr[min\_index] *# predict the label of the nearest example*

注：L1和L2距离比较，L2会放大较大差异，故适用于中等差异。

**k-Nearest Neighbor分类器**

1.与其只找最相近的那1个图片的标签，我们找最相似的k个图片的标签，然后让他们针对测试图片进行投票，最后把票数最高的标签作为对测试图片的预测。



异常数据点（蓝色区域中绿点）在NN中产生错误预测，在KNN中都被平滑了，灰色区域说明近邻标签的票数相同

2.如何选择k：

1）L1L2，k，点积这些选择称为超参数，测试集非常珍贵，不能用来优化参数，会引起测试集过拟合

测试数据只能使用一次→从50000训练集中拿出1000作为验证集→代入不同的k求准确率均值

2）交叉验证：用于数量级较小，5000分成5份，1份用来验证，其余4份循环训练（70~90%），取所有5次验证结果的均值作为验证结果

注：1、如果超参数多，需要更多验证集，一般选用交叉验证

2、不要把验证集重新放回训练集，这样有可能会破坏对最优参数的估计，直接使用测试集测试最优模型，得到KNN的性能表现。

NN的优缺点：

1、在实际应用中更在乎测试效率，它的特性与卷积相反，卷积花费大量时间训练，测试很快

2、ANN可以通过牺牲准确率提升计算速度，通常依赖一个预处理/索引过程，这个过程包含kd树的建立和k-means算法的应用

3、在高维度数据上，基于像素的距离和感官上的非常不同。另外，相近图片一般是被背景主导而非语义本身

反向传播

1.函数关于每个变量的导数表示对该变量的敏感程度

2.前向传播是wx+b的计算传播，反向传播是梯度传播，每个门单元将回传的梯度乘以对自身输入的局部梯度，从而得到整个网络的输出对该门单元的每个输入值的梯度。

3.神经网络中最常用的加法、乘法和取最大值这三个门单元。加法将梯度相等地分发给它的输入，取最大将梯度交给更大的输入，乘法门对他们进行交换（乘法门的局部梯度就是输入值，如果x变大，w也会变大，这样必须降低学习率来弥补，所以数据预处理很重要），然后乘以梯度

4.在矩阵运算中，注意分析维度和转置，dw的维度和w是一样的。

神经网络笔记1

1. 单个神经元可以用来实现一个二分类分类器。在SVM/Softmax的例子中，正则化损失从生物学角度可以看做逐渐遗忘，因为它的效果是让所有突触权重w在参数更新过程中逐渐向着0变化。
2. 常用激活函数

1）Sigmoid [0,1] >>>(1-fx)fx tanh [-1,1] >>>1-x\*\*2

都存在饱和段梯度消失的问题

2）Relu 收敛快，当一个很大的梯度流过ReLU的神经元的时候，output=0，梯度为0，与该神经元有关的w将不会变化，通过合理设置学习率，这种情况的发生概率会降低。

注：一般使用relu，死亡率高改用leaky relu

1. 神经网络结构
2. 命名规则。当我们说N层神经网络的时候，我们没有把输入层算入。ANN/MLP（多层感知器）来指代神经网络，也有人用单元（unit）而不是神经元作为术语
3. 输出层的神经元一般是不会有激活函数，因为是特定实数
4. 神经网络的尺寸的标准主要有两个：一个是神经元的个数，另一个是参数的个数
5. 每个神经元的权重都在W1的一个行中，于是矩阵乘法np.dot(W1, x)就能计算该层中所有神经元的激活数据。每个输入样本将会是x中的一列，所有的样本将会被并行化的高效计算出来。
6. 表达能力

神经网络是一个通用函数近似器，在实践中3层的神经网络会比2层的表现好，然而继续加深（做到4，5，6层）很少有太大帮助。卷积神经网络的情况却不同，深度是一个极端重要的因素(以10为量级个可学习的层）。对于该现象的解释是：因为图像拥有层次化结构（比如脸是由眼睛等组成，眼睛又是由边缘组成），所以多层处理对于这种数据就有直观意义。

1. 设置层的数量和尺寸

神经元过多，过拟合（Overfitting）对数据中的噪声有很强的拟合能力。但是数据量少时，应采取控制过拟合而非减少神经元数目。

一个小网络，某些情况下运气好会收敛到一个好的地方，某些情况下就收敛到一个不好的极值。大的网络最终损失值的差异将会小很多，比较稳定。

神经网络笔记2

一、数据预处理：数据预处理有3个常用的符号，数据矩阵X，假设其尺寸是[N x D]（N是数据样本的数量，D是数据的维度）。

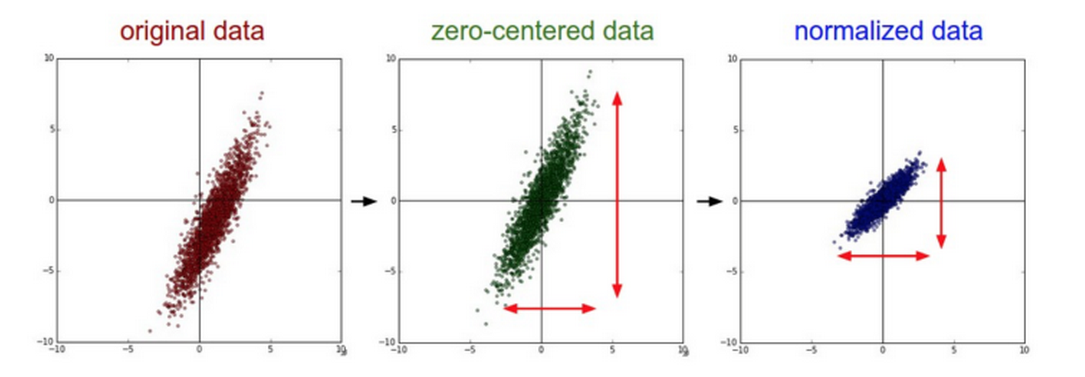
1. 均值减法

1）若是对每个独立特征减去平均值 x-=np.mean(x,axis=0)

2）对于图像一般是每个像素都减去一个值 x-=np.mean(x)

1. 归一化：将数据所有维度归一化。适用于每个维度数值范围/单位不同

1）先对数据做零中心化，再每个维度除以标准差x/=np.std(x,axis=0)



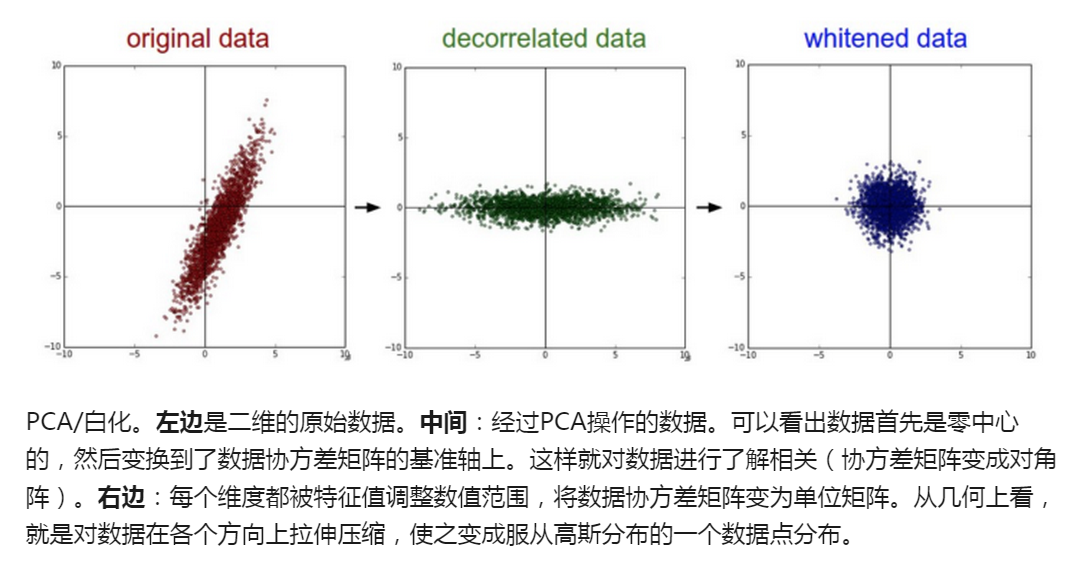
2）每个维度做归一化，-1~1

3、PCA和白化 （为什么神经网络一般不用）

先对数据进行零中心化处理，然后计算协方差矩阵

X-=np.mean(x,axis=0)

cov=np.dot(x.T,x)/x.shape[0]



注：预处理只从训练集中求图片平均值，然后再用（验证、测试集）中的图像减去这个平均值

二、权重初始化

1）小随机数初始化 W = 0.01 \* np.random.randn(D,H)

权重初始值要非常接近0又不能等于0，但并不是小数值一定会得到好的结果，因为可能会梯度过小

2）使用1/sqrt(n)校准方差 w = np.random.randn(n) / sqrt(n) 其中n是输入数据的数量，使得神经元输出有和输入x一样的方差

Var(∑wx)=(nVar(w))Var(x)

3）使用ReLU神经元的当前最佳 网络中神经元的方差应该是2.0/n

w = np.random.randn(n) \* sqrt(2.0/n) b初始化一般为0

4）批量归一化（Batch Normalization）全连接层（或者是卷积层）与激活函数之间添加一个BatchNorm层，使激活数据符合高斯分布。使用了批量归一化的网络对于不好的初始值有更强的鲁棒性。

Python

三、正则化 Regularization

L2正则化

卷积神经网络

一、卷积神经网络主要由三种类型的层构成：卷积层，汇聚（Pooling）层和全连接层。

有的层有参数，有的没有（卷积层和全连接层有，ReLU层和汇聚层没有）。

有的层有额外的超参数，有的没有（卷积层、全连接层和汇聚层有，ReLU层没有）。

二、卷积层：3个超参数控制着输出数据体的尺寸：

1）深度（depth）: 它和使用的滤波器的数量一致，而每个滤波器在输入数据中寻找一些不同的东西，将沿着深度方向排列、感受野相同的神经元集合称为深度列（depth column）/纤维（fibre）

步长（stride）:1/2

零填充（zero-padding）：最常用的是用来保持输入数据体在空间上的尺寸，这样输入和输出的宽高都相等

2）卷积计算：对应位置相乘求和