**课堂笔记**

1. **KNN**
2. 总括

KNN是一个图像分类算法，它使用某种距离指标衡量两幅图像之间的相似度。当判断一个新的图像x的类别时，会将x和所有已知类别的图像比较相似度，使用最相似的k张图像进行投票，从而确定x的类别。

1. 交叉验证

将训练集分成K份，对每一个超参数，循环选取K-1份用作训练集，1份用作验证集。此时，对于每一个超参数将会得到K个准确率，取其均值作为其准确率。

个人理解：交叉验证使训练集中每个数据都有作为验证集的机会，因此在需要大量验证集（可能超参数数量较多，因此需要数量较多的验证集）会比较有用

1. 优劣势

优势：算法简单

劣势：训练时间短，计算时间长，不利于实时；像素距离与感官距离有很大不同，因此准确率不高；图片分类由背景主导，而不是景物（背景所占像素空间较大）

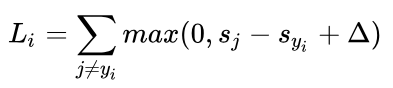
1. **线性分类器**
2. 总体算法

使用一个评分函数，将图像的像素值映射为各个分类类别的得分；使用一个损失函数，用来量化预测分类标签的得分与真实标签之间一致性。因此可将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。

1. 线性分类器的映射函数



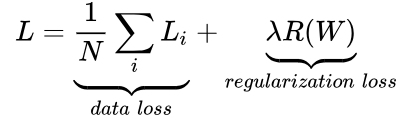
1. 损失函数
2. 多分类SVM损失



Max(0,-)称为折页损失

多类SVM“想要”正确类别的分类分数比其他不正确分类类别的分数要高，而且至少高出delta的边界值。

正则化：满足最小Li的W可能有很多(aW都是)，因此向某些特定的权重**W**添加一些偏好，对其他权重则不添加，以此来消除模糊性。我的理解是，可以让某些权重值大一些；某些权重值小一点，而不是给权重更多的自由，以此防止过拟合。L2惩罚倾向于更小更分散的权重向量，这就会鼓励分类器最终将所有维度上的特征都用起来，而不是强烈依赖其中少数几个维度。在后面的课程中可以看到，这一效果将会提升分类器的泛化能力，并避免*过拟合*。最常用的正则化惩罚是L2范式，损失函数就变为下面的形式，



1. Softmax分类器

其损失函数使用交叉熵损失



从概率论的角度来理解，我们就是在最小化正确分类的负对数概率，这可以看做是在进行*最大似然估计；从信息论角度来看，交叉熵损失函数“想要”预测分布的所有概率密度都在正确分类上。*

1. **最优化**
2. **反向传播**
3. **神经网络**
4. 非线性函数的引入（激活层）

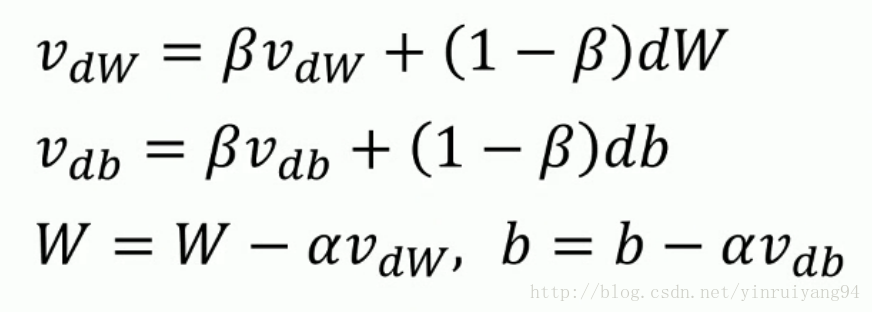
如果不引用非线性函数，用再多的参数W1,W2,W3,最后都会变成某个W\*x(W=W1\*W2\*W3)，仍然是个线性函数，本质上并没有改变

1. 神经元

一个单独的神经元可以用来实现一个二分类分类器，比如二分类的Softmax或者SVM分类器。

3、随机梯度下降及各种更新方法

1）动量更新β是可以自行设置的超参数，一般情况下默认为0.9



可以看到v是之前的梯度的加权叠加，因此作出当前步长的决定不仅和当前的梯度相关，还和之前一段时间梯度相关。

举个例子，如果你站在一个地方不动，让你立刻向后转齐步走，你可以迅速向后转然后就向相反的方向走了起来，批梯度下降和随机梯度下降就是这样，某一时刻的梯度只与这一时刻有关，改变方向可以做到立刻就变。而如果你正在按照某个速度向前跑，再让你立刻向后转，可以想象得到吧，此时你无法立刻将速度降为0然后改变方向，你由于之前的速度的作用，有可能会慢慢减速然后转一个弯。动量梯度下降是同理的，每一次梯度下降都会有一个之前的速度的作用，如果我这次的方向与之前相同，则会因为之前的速度继续加速；如果这次的方向与之前相反，则会由于之前存在速度的作用不会产生一个急转弯，而是尽量把路线向一条直线拉过去。

这就解决了文中第一个图的那个在普通梯度下降中存在的下降路线折来折去浪费时间的问题。

2）Nesterov动量

4、**逐参数适应学习率方法**

1）Adagrad算法

随着迭代次数的增加学习率降低，缺点是，在Adagrad算法中，不断有正数加到分母的cache变量中，步长就会逐渐衰减到0，最后完全停止学习。在深度学习中单调的学习率被证明通常过于激进且过早停止学习。

当我们在垂直方向上看到许多大的梯度，Adagrad就会衰减学习速率，使垂直方向的更新步长越来越小。在水平方向上的梯度是很小的，所以分母会变小，相比于垂直方向，水平方向更新更快。这就是对每个参数自适应不同的学习速率，针对不同梯度方向的补偿措施。

2）RMSprop

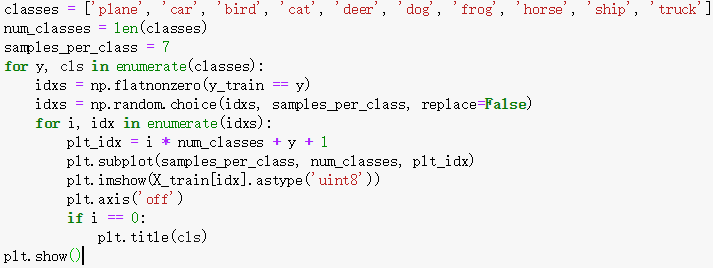
用一种很简单的方式修改了Adagrad方法，让它不那么激进，单调地降低了学习率。

3）Adam

像是RMSProp的动量版

**Assignment1**

1. 可视化数据集中部分数据



1. 计算准确度

Accuracy=np.sum(y\_pred==y\_val)/y\_val.shape[0]

print ('k = %d, accuracy = %f' % (k, accuracy))

1. numpy广播

发生条件：当两个矩阵维度相同，或者任意一个维度是1。Eg.n\*1+n\*m=n\*m

应用场景：可以用广播和其它numpy函数实现向量化运算

1. numpy常用函数

argsort(a, axis=-1, kind='quicksort', order=None)Returns the indices that would sort an array，将a升序排序，返回排序后的index

l=np.bincount(x)#bincount统计x中各个元素（非负整数）出现的次数，返回次数列表。返回列表中下标i的值表示i在x中出现的次数

np.argmax(l)#返回l中最大值

X\_num=np.array\_split(X,num)#将X分为num份，X\_num[i]表示第i份

Np.sum(x,axis=1)按行求和

1. 模型常见写法

Class LinearClassifier(object):

def \_\_init\_\_(self):

self.W = None

def train(self, X, y, learning\_rate=1e-3, reg=1e-5, num\_iters=100,

batch\_size=200, verbose=False):

def predict(self, X):

def loss(self, X\_batch, y\_batch, reg):

1. continue#结束当前的循序，进行下一个数的循环,break是结束循环
2. {}表示字典，[]表示列表，()表示元组