>一.何为Tensorflow与深度学习

1.Tensorflow

Tensor是张量，可以理解为任意一个维度的矩阵

flow就是流动，也就是说，在矩阵中流动，它是用来进行GPU加速的矩阵

Tensorflow是一个非常强大的深度学习框架

2.深度学习

机器学习的流程:

- 数据获取
- **特征工程**
- 建立模型
- 评估与应用

最为重要的就是特征工程

特征工程的作用:

- 数据特征决定了模型的上限
- 预处理和特征提取是最核心的
- 算法与参数选择决定了如何逼近这个上限

可以把深度学习当做一个黑盒子，比如在数据处理中，它会对原始数据做各种各样的变化操作，进行自动的特征提取。它可以告诉我们什么特征是最合适的。我对这个的理解就是，这个就是一个进行复杂运算的函数。

传统人工智能算法和深度学习算法如果想要看出效果上的区别，最重要的就是**数据的规模！**

二.深度学习在计算机视觉上的应用

1.图像分类

图像的表示

一张正常的彩色图被表示为三维数组的形式，即横纵坐标还有颜色通道，从0-255.

2.计算机视觉面临的挑战

- 照射角度
- 形状改变
- 部分遮蔽
- 背景混入

3.机器学习的常规套路:

- 收集数据并且给定标签
- 训练一个分类器
- 测试评估

表示成代码可以是这样的

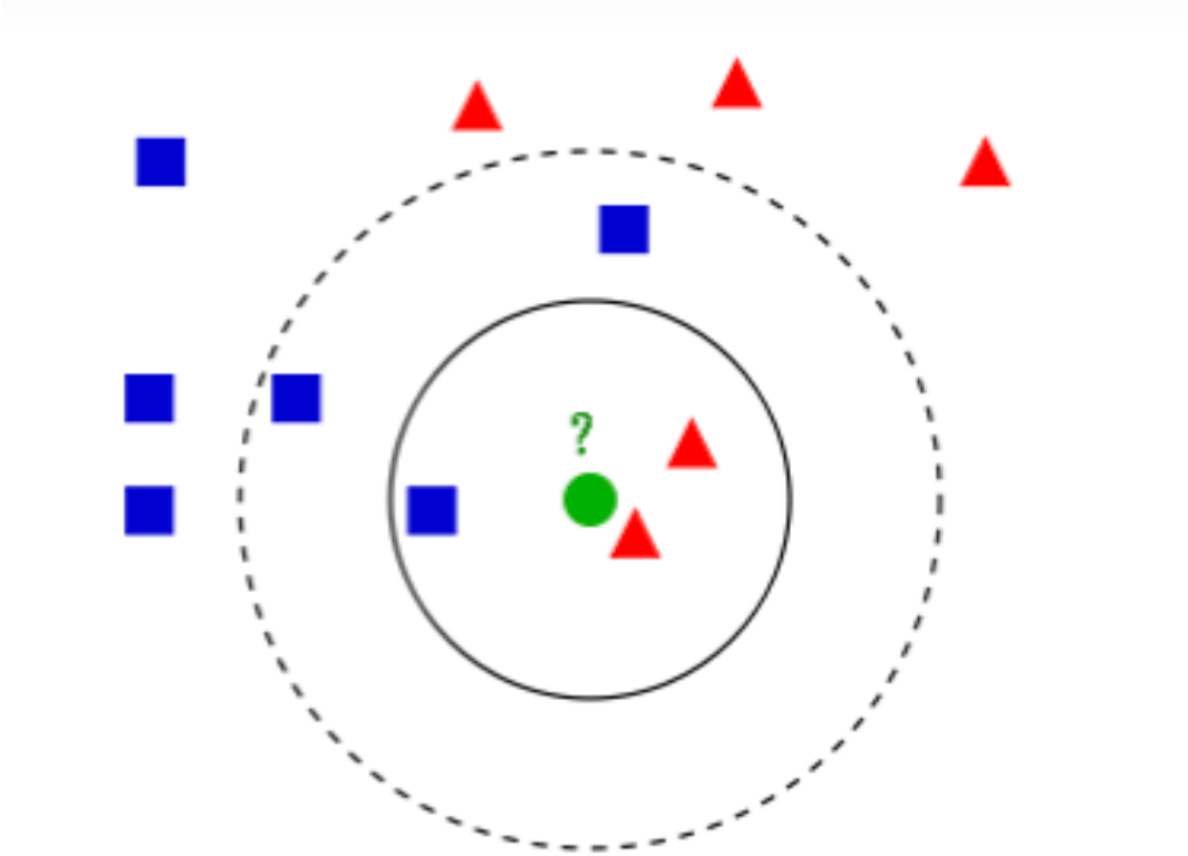
```
def train(train_images,train_labels)
    #build a model for image->lables...
    return model

def predict(model,test_images)
    #predict test labels using the model
    return test_labels
```

4.深度学习的应用算法

K近邻算法

K近邻算法是一个简单的机器学习的分类方法，基于距离和权重



这是一张图像，里面绿色圆表示未知形状。我们以这个绿色圆为圆心画一个圆，这个时候有一个正方形两个三角形，那么此时就是3个最近邻的点，也就是说，设一个k为离得最近的点，此时k=3,三角形比正方形多，那么这个时候认定这个绿色圆为三角形。然后再画一个更大的圆，选取出了五个最近邻的点，此时正方形多，也就是说这个时候认定这个绿色圆为正方形。(关于为什么K都是奇数，我个人的观点是和卷积核差不多，如果不是奇数卷积核矩阵就没有中心点了，也不对称了，这里不是奇数的话会出现个数相等的情况导致无法判定究竟是哪个了)

K近邻的计算流程

- 计算已知类别数据集中的点与当前点的距离
- 按照距离依次排序
- 选取与当前点距离最小的K个点
- 确定前K个点所在类别的出现概率
- 返回前K个点出现频率最高的类别作为当前点预测分类

K近邻分析

- KNN算法本身简单有效，属于lazy_learning算法
- 分类器不需要用训练集来训练，训练时间复杂度是0
- KNN分类的计算复杂度和训练集中的文档书目成正比，也就是说，如果训练集中文档总数是n，那么KNN分类时间复杂度是O(n)
- k值的选择，距离度量和分类决策规则是该算法的三个基本要素

用K近邻进行数据分类

用***CIFAR-10***的数据集(10类标签，50000个训练数据，10000个测试数据，大小均为32*32)进行图像分类任务

- **距离的选择**

  用测试数据的矩阵和训练数据的矩阵进行相减操作(就是像素点对应相减)，得到的最后矩阵，比如一个4*4的矩阵，将其16个元素相加得到的就是图像距离

- **进行K近邻分类**

  对距离进行排序之后，每个类别选出最近的几张图像

最终测试结果

- 部分结果还是匹配很好的
- 部分结果出现了错误，没有分类正确，出现背景相同的被放到了一起，造成的原因是没有告知哪一块是主体，哪一块是背景

最终结论：K近邻不能用来进行图像分类

- 背景主导是一个最大的问题，我们关心的是主体(主要成分)
- 我们必须让机器认知到哪些是重要的成分才能很好地解决这个问题

二.神经网络基础

1.线性函数

从输入-->输出的映射

用得分公式进行计算

$$score = f(images, parameters)$$

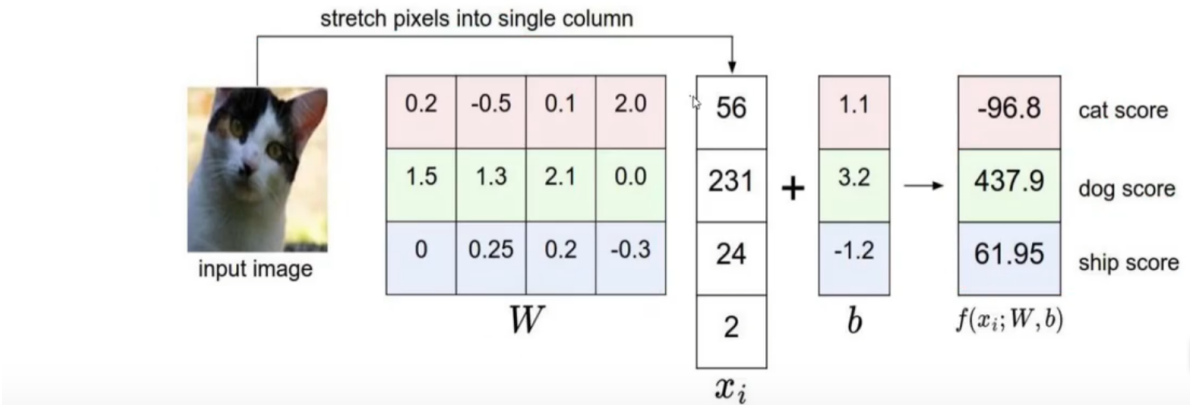
这样就可以计算出某一个类别的得分。我们需要计算的是一张图像在所有类别的得分。每个像素点的权重是不一样的，有的是促进性的，有的是抑制性的。比如32x32x3的图片，就需要3072个权重参数

用x表示images,用W表示权重。

$$f(x, W) = Wx(+b)$$

这个b是偏置参数，数值上等于所有的类别的个数，起到一个微调作用，比如猫类狗类。

计算方法



可以看到图里面的结果是不正确的，猫的得分是负分，而狗的类别的分数很高。这个W里面的值会对结果产生决定性影响。这个W参数一开始是随机的值，后面会一直迭代，去改进这个W参数。

2.损失函数

损失函数有很多种

$$Li = \Sigma_{j \neq i} max(0, s_j - s_{yi} + 1)$$

这个sj表示的是不正确类别，这个syi表示的是正确的类别，比如一共有N个类别，那么这个式子就要计算N-1次，然后把各自的max相加得到最终结果。

损失函数的值越小说明匹配的越好，反之说明错误得越大

- 损失函数=数据损失+正则化惩罚项
- $$L = \frac{1}{N} * \Sigma_{i=1}^{N} * max(0, f(x_i; W)j - f(x_i; w)y_i + 1) + \lambda R(W)$$
- 正则化惩罚项
- $$R(W) = \Sigma_k \Sigma_l W_{k,l}^2$$

我们总是希望模型不要太过复杂，过拟合的模型是没用的

3.前向传播

Softmax分类器

我们要将前面获得的得分值转换为0-1的一个概率值

$$g(z) = \frac{1}{1+e^{-z}}$$

比如说得分分别是3.2分，5.1分，-1.7分现对其做一个exp运算，也就是

$$e^x$$

然后要对其做一个归一化操作

此时这个值是24.5,164.0,0.18

我们用他们的值去比上他们的总和，就是归一化之后的概率了，最终结果是0.13,0.87,0.00

概率值越接近1的，表示损失越小。

得出损失值

- 归一化公式可表示为

$$P(Y=k|X=x_i) = \frac{e^s k}{\Sigma_j e^s j} \quad where \quad s=f(x_i;W)$$

- 计算损失值

$$Li = -logP(Y=y_i|X=x_i)$$

比如这个Li是0.13，那么它的损失值就是-log(0.13),计算得出是0.89

回归任务是由得分值计算损失，分类任务由概率计算损失

我们的核心其实就是通过调节W来减少loss值

一般来说，前向传播是要经过几层变换的，也就是对原始数据的多个部分进行多个权重参数的计算

我们利用反向传播(梯度下降)来更新模型

4.梯度下降

梯度下降是一种优化算法，目的是求出目标函数的最优解。

- 假设有目标函数：

$$J(\theta_0,\theta_1) = \frac{1}{2m}\Sigma_{i=1}^{m}(h_\theta(x^{(i)}-y^{(i)}))$$

- 寻找山谷的最低点，也就是我们的目标函数终点(什么样的参数能使得目标函数达到极值点)
- 更新参数

    - 1.找到当前最合适的方向
    - 2.每次走一小步
    - 按照方向与步伐去更新参数
- 批量梯度下降容易得到最优解，但是由于每次考虑所有样本，导致速度很慢
- 随机梯度下降每次找一个样本，迭代速度快，但不一定每次都朝着收敛的方向
- 小批量梯度下降法每次更新选择一小部分数据来计算，比较实用

5.反向传播

比如说原始数据需要进行三次变换：

$$f = ((xW1)xW2)xW3$$

我们先要看W3对后面做了多大贡献(这是一个求偏导的过程)，然后是W2，W1

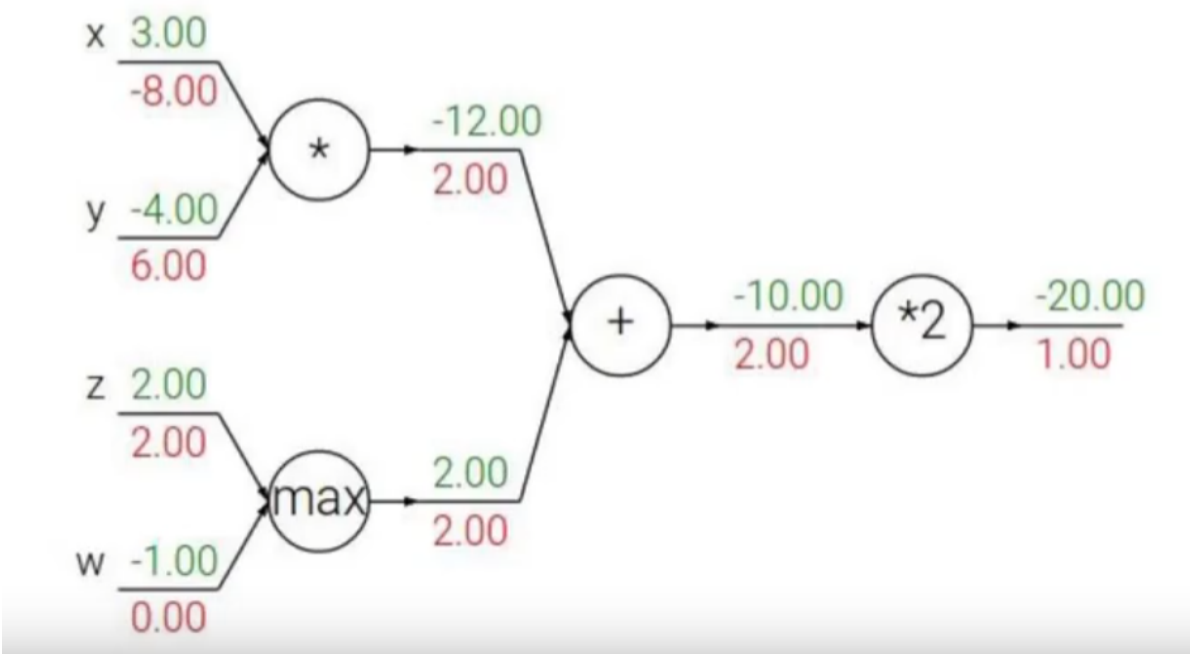
我们有这样一个从后往前计算的顺序。

链式法则

- 梯度是一步一步传播的，

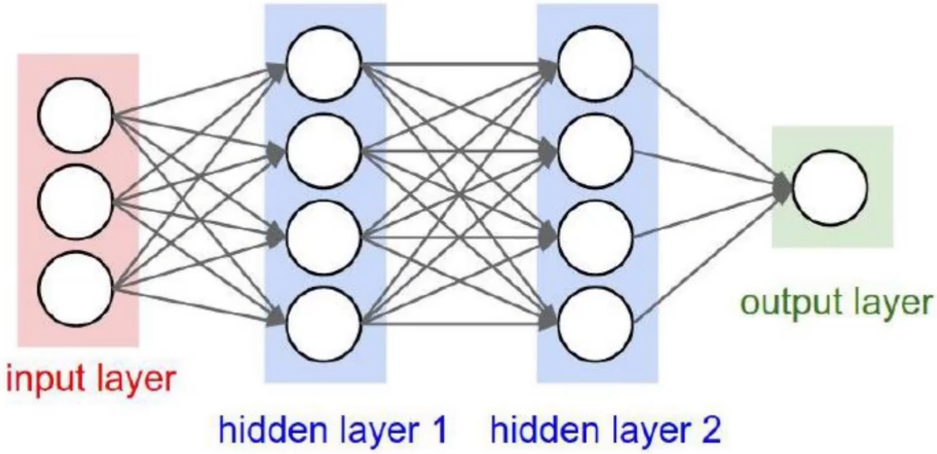
门单元

- 加法门单元：均等分配
- Max门单元：给最大的
- 乘法门单元：互换的感觉



6.整体架构

- 整体架构
  - 层次结构
  - 神经元
  - 全连接
  - 非线性



- 基本机构

$$f = W_2 max(0, W_1 x)$$

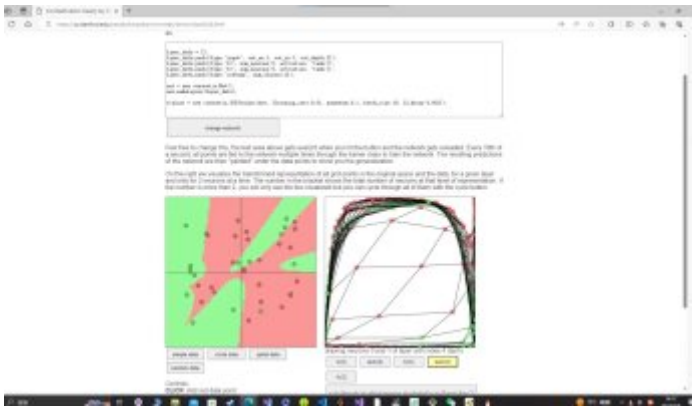
- 继续堆叠一层:

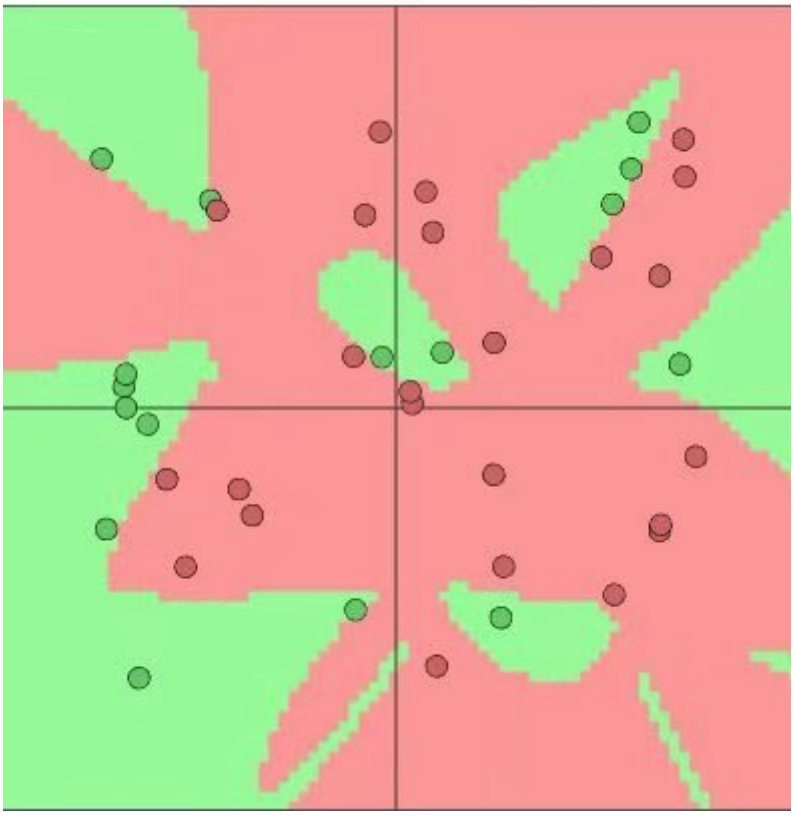
$$f = W_3 max(0, W_2 max(0, W_1 x))$$

- 神经网络的强大之处在于，用更多的参数来拟合复杂的数据

神经元个数越多，过拟合程度越高，得到的效果可能会越好，但是速度会比较慢

当然，个数过多会导致过拟合。





前面是五个神经元做的神经网络模型，后面是十个神经元做的，后者的计算量要远远超出前者。看似后面的十个神经元做的模型分毫不差，实际上有很大的问题。这两张图最主要的区别就是右下角的那个绿色点，五个神经元模型的没有把它包括在内，十个神经元模型的却把它框选起来了。按照K近邻算法说的话，如果在这个右下角的附近选取一个点，那它会被判定成红色，也就是说，这一块的区域实际上大概率应该是红色的，然而模型上给出的是一整片的绿色，这就是过拟合。