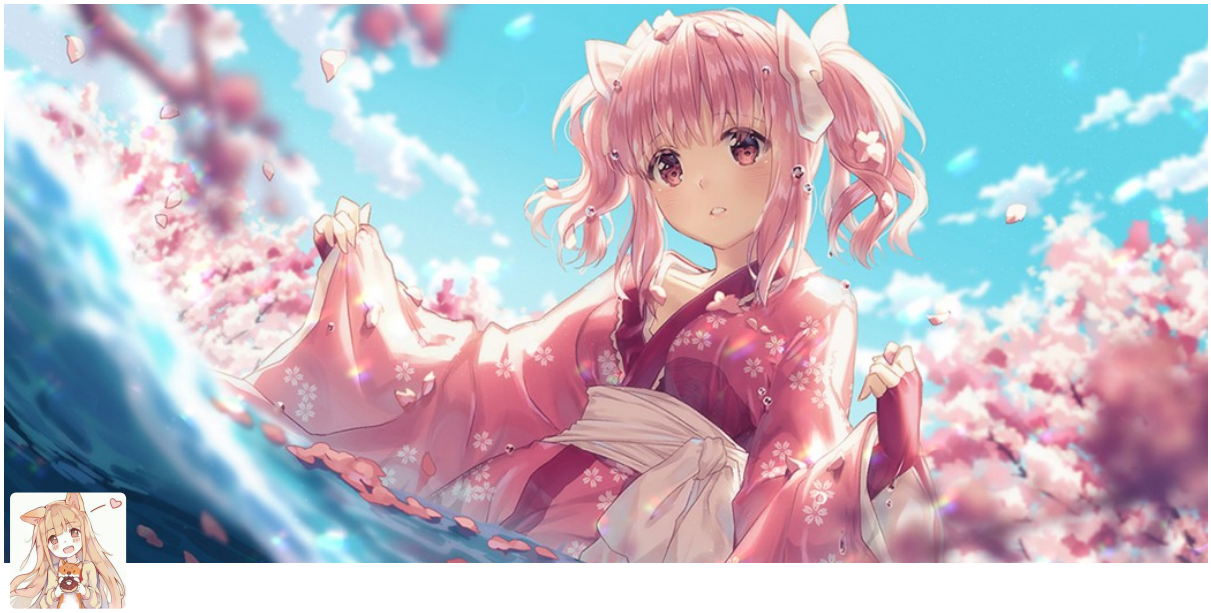


卷积神经网络基础及AlexNet网络搭建

数据预处理操作：

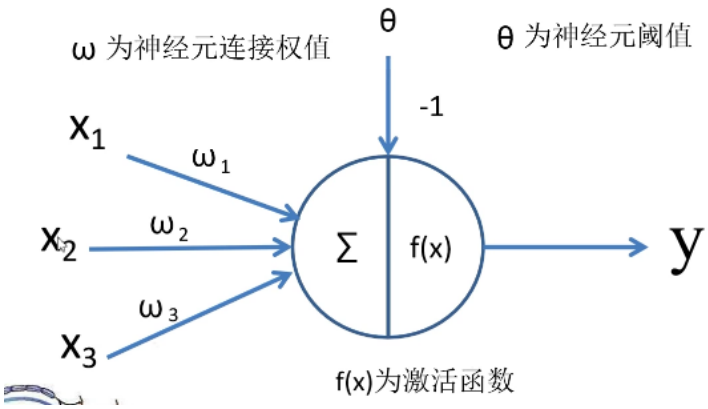
1.随机剪裁

2.翻转

3.转换成张量

4.归一化

全连接层：



将激励乘以对应的权重再相加：

$$y = f(x_1 \cdot \omega_1 + x_2 \cdot \omega_2 + x_3 \cdot \omega_3 - 1)$$

BP算法：

包括信号的前向传播和误差的反向传播两个过程。

反向更新：

当我们算出y值以后，带入相应的激活函数（交叉熵），

针对多分类问题(softmax输出，所有输出概率和为1）：

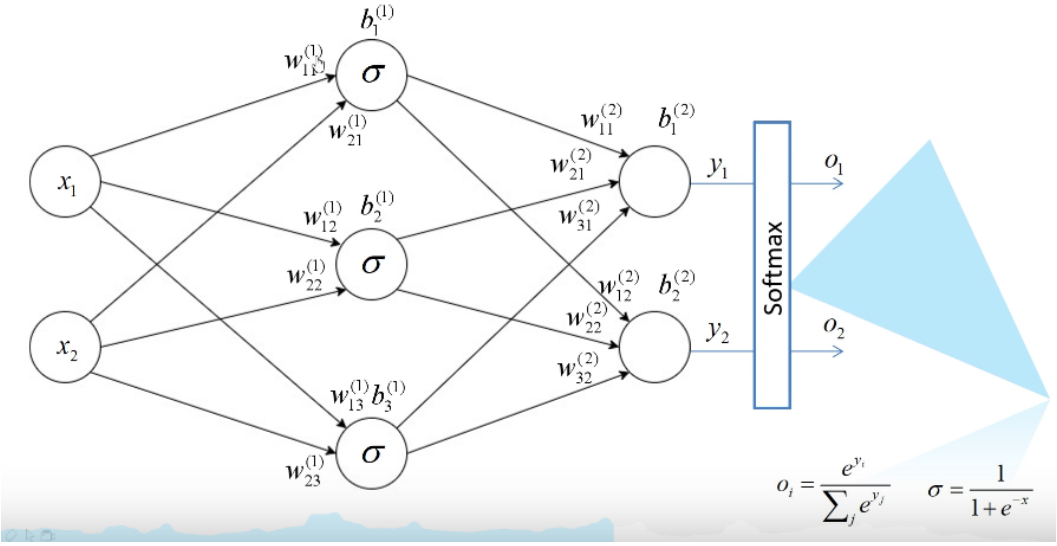
$$H = -\sum_i o_i^* \log(o_i)$$

对于二分类问题：

$$H = -\frac{1}{N}\sum_{i=1}^{N}[o_i^* \log o_i + (1-o_i^*)\log(1-o_i)]$$

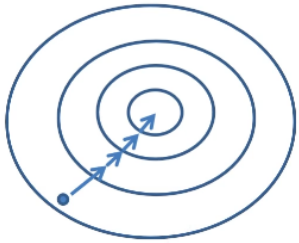
其中，o*为真实标签值，o为预测值，默认log以ln为底。

算出对应的o：

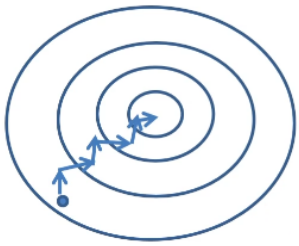


算出o后，再带入损失函数算出损失，再与W11等相加，实现反向传播。

注意，此时我们还要注意损失梯度的方向：



若使用整个样本集进行求解
损失梯度指向全局最优方向



若使用分批次样本进行求解
损失梯度指向当前批次最优方向

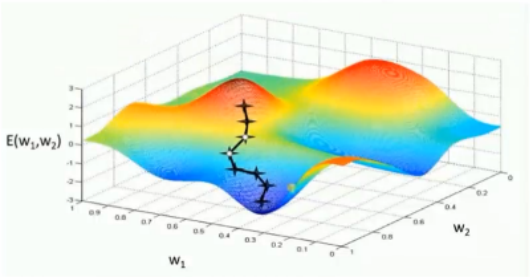
由于设备等原因的限制，所以我们常常采用第二种分批次求解损失梯度指向当前批次最优方向。

以上操作就是我们说的SGD优化器优化器。

优化器：

目的：使网络更快的收敛。

（1）.SGD优化器



$$w_{t+1} = w_t - \alpha \cdot g(w_t)$$

α为学习率，$g(w_t)$为t 时刻对参数w_t的损失梯度

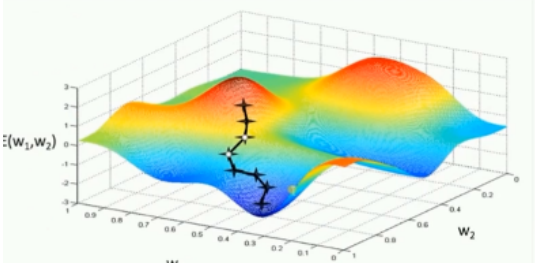
缺点：

易受样本噪声的影响

可能陷入局部最优解

为了解决上述的缺点，我们引入了第二种优化器。

（2）SGD+Momentum优化器



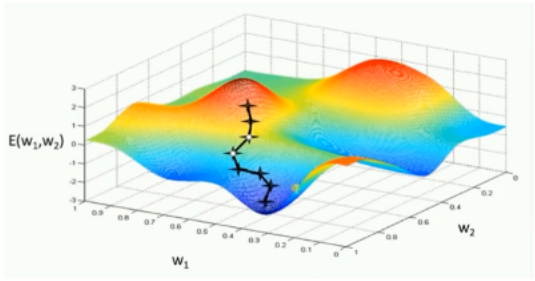
$$v_t = \eta \cdot v_{t-1} + \alpha \cdot g(w_t)$$
$$w_{t+1} = w_t - v_t$$

α为学习率，$g(w_t)$为t 时刻对参数w_t的损失梯度

$\eta(0.9)$为动量系数

引入了懂了概念。

（3）.Adagrad优化器（自适应学习率）



$$s_t = s_{t-1} + g(w_t)\cdot g(w_t)$$

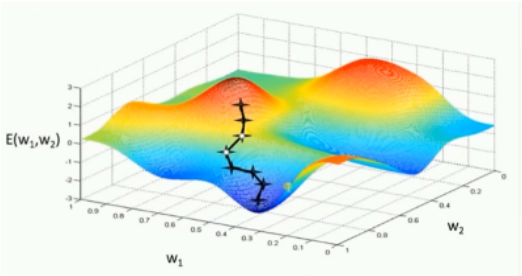
$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{s_t+\varepsilon}}\cdot g(w_t)$$

α为学习率，$g(w_t)$为t时刻对参数w_t的损失梯度

$\varepsilon(10^{-7})$为防止分母为零的小数

缺点：学习率下降太快可能还没收敛就停止了。

（4）.RMSProp优化器（自适应学习率）



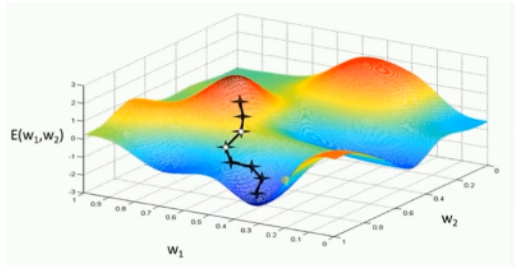
$$s_t = \eta\cdot s_{t-1} + (1-\eta)\cdot g(w_t)\cdot g(w_t)$$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{s_t+\varepsilon}}\cdot g(w_t)$$

α为学习率，$g(w_t)$为t时刻对参数w_t的损失梯度

$\eta(0.9)$控制衰减速度，$\varepsilon(10^{-7})$为防止分母为零的小数

（5）.Adam优化器



$$m_t = \beta_1\cdot m_{t-1} + (1-\beta_1)\cdot g(w_t) \quad \text{一阶动量}$$

$$v_t = \beta_2\cdot v_{t-1} + (1-\beta_2)\cdot g(w_t)\cdot g(w_t) \quad \text{二阶动量}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad \hat{v}_t = \frac{v_t}{1-\beta_2^t}$$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{v}_t+\varepsilon}}\hat{m}_t$$

α为学习率，$g(w_t)$为t时刻对参数w_t的损失梯度

$\beta_1(0.9)$、$\beta_2(0.999)$控制衰减速度，$\varepsilon(10^{-7})$为防止分母为零的小数

one-hot编码：

1	0	0	0	0	0	0	0	0	0	→ 0
0	1	0	0	0	0	0	0	0	0	→ 1
0	0	1	0	0	0	0	0	0	0	→ 2
0	0	0	1	0	0	0	0	0	0	→ 3
0	0	0	0	1	0	0	0	0	0	→ 4
0	0	0	0	0	1	0	0	0	0	→ 5
0	0	0	0	0	0	1	0	0	0	→ 6
0	0	0	0	0	0	0	1	0	0	→ 7
0	0	0	0	0	0	0	0	1	0	→ 8
0	0	0	0	0	0	0	0	0	1	→ 9

卷积层：

卷积的特性：

1.拥有局部感知机制

2.权值共享

卷积操作：通过滑动的卷积核，将被覆盖的特征图中的值与卷积核中的权重相乘再相加，就可以得到对应的卷积结果。

卷积目的：进行图像的特征提取

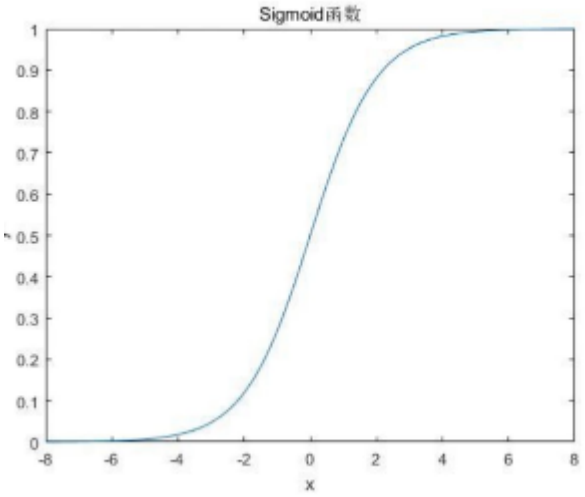
注意：

1.卷积核的维度与输入特征图维度相同

2.输出的特征矩阵channel与卷积核个数相同

3.加偏执只要矩阵中所有元素加上偏执就可以

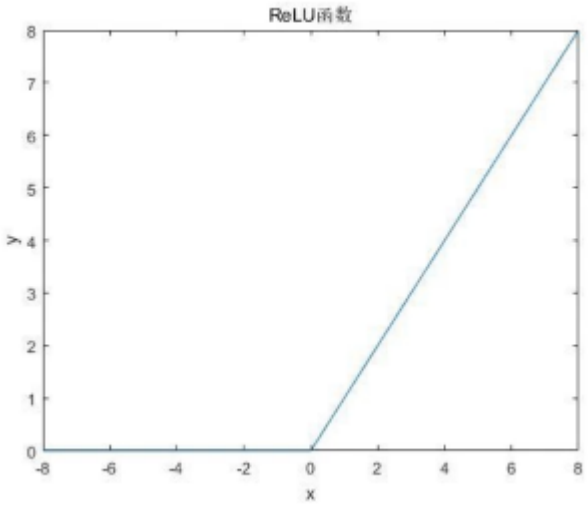
激活函数：



sigmoid激活函数

$$f(x)=\frac{1}{1+e^{-x}}$$

sigmoid激活函数饱和时梯度值非常小，故网络层数较深时易出现梯度消失。



Relu激活函数

$$f(x)=Max(0,x)$$

缺点在于当反向传播过程中有一个非常大的梯度经过时，反向传播更新后可能导致权重分布中心小于零，导致该处的倒数始终为o，反向传播无法更新权重，即进入失活状态。

在卷积操作过程中，矩阵经卷积操作后的尺寸由以下几个因数决定:

1	0	2	5
2	3	4	9
1	7	8	5
3	3	1	5

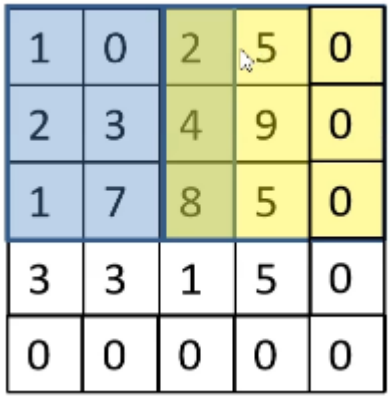
1.输入图片大小W×W

2.Filter大小F×F

3.步长S

4.padding的像素值P

很显然，如果我们对上图的特征图进行3×3步长为2的卷积，那一定会出现越界，所以我们在越界部分补零：



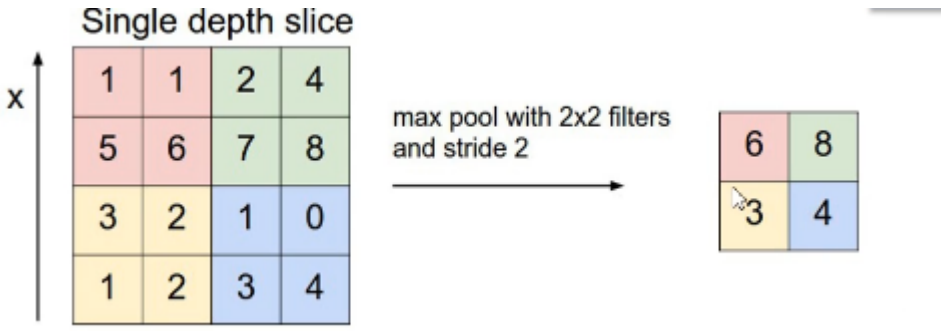
经卷积后的矩阵尺寸大小计算公式为：

$$N = (W - F + 2P) / S + 1$$

池化层：

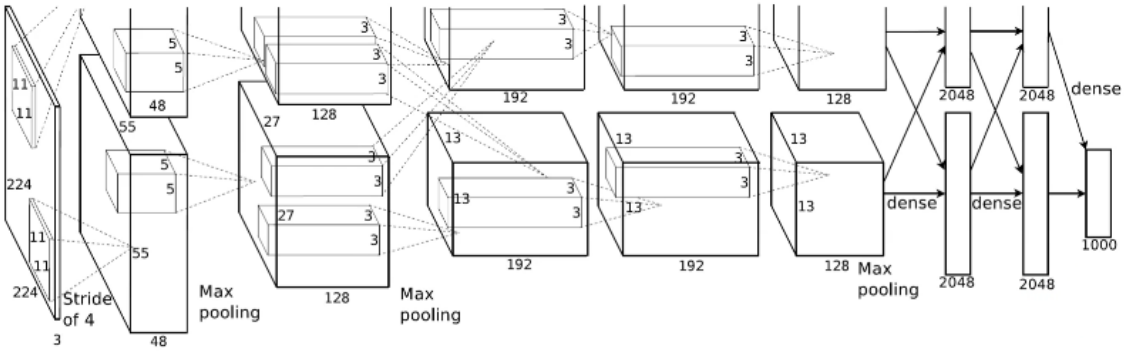
目的：对特征图进行稀疏处理，减少数据的运算量。

最大池化：



AlexNet神经网络：

网络结构：



Conv1:	Maxpool1:	Conv2:	Maxpool2:	Conv3:	Conv4:	Conv5:
kernels:96 kernel_size:11 padding: [1, 2] stride:4	kernel_size:3 pading: 0 stride:2	kernels:256 kernel_size:5 padding: [2, 2] stride:1	kernel_size:3 pading: 0 stride:2	kernels:=384 kernel_size:3 padding: [1, 1] stride:1	kernels:384 kernel_size:3 padding: [1, 1] stride:1	kernels:128*2=256 kernel_size:3 padding: [1, 1] stride:1
output_size: [55, 55, 96]	output_size: [27, 27, 96]	output_size: [27, 27, 256]	output_size: [13, 13, 256]	output_size: [13, 13, 384]	output_size: [13, 13, 384]	output_size: [13, 13, 256]

特点：

1.首次利川GPU进行网络加速训练。

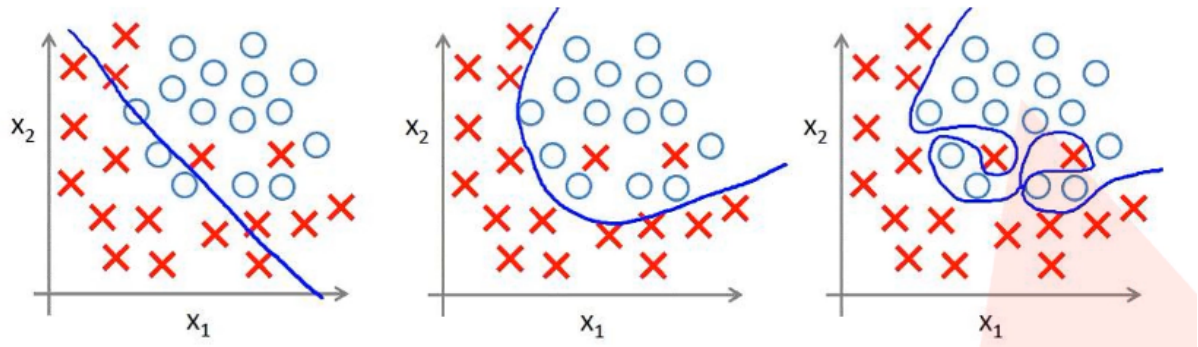
2.使用了 ReLU激活函数，而不是传统的Sigmoid激活函数以及Tanh激活函数。

3.使用了LRN局部响应归一化。

4.在全连接层的前两层中使用了Dropout随机失活神经元操作，以减少过拟合。

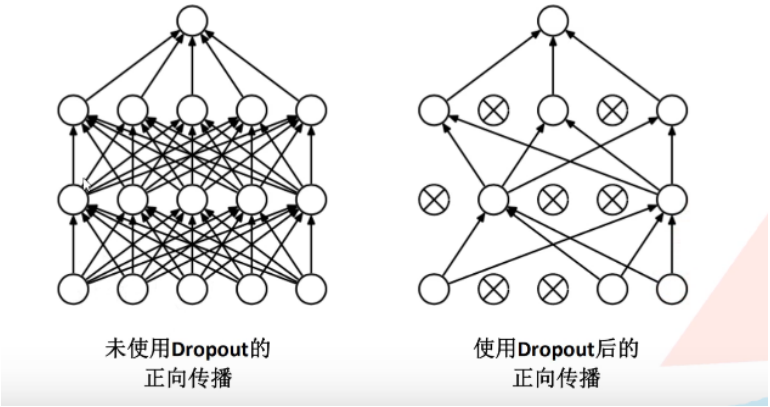
过拟合:

根本原因是特征维度过多，模型假设过于复杂，参数过多，训练数据过少，噪声过多，导致拟合的函数完美的预测训练集，但对新数据的测试集预测结果差。过度的拟合了训练数据，而没有考虑到泛化能力。



怎么解决过拟合的问题呢？

1.使用Dropout的方式在网络正向传播过程中随机失活一部分神经元。



2.图像预处理随机剪裁完以后，进行翻转进行数据增强

权重、偏置的初始化：

为什么要对权重和偏执进行初始化呢？

好的初始化权重、偏置能加快网络的收敛性，坏的初始化参数在影响网络收敛性的同时还会引起网络产生对应的梯度问题（过大的权重参数会导致梯度爆炸、过小引起梯度消失，或者收敛至局部最小值）

而AlexNet神经网络，采取的是Kaiming初始化的方法。