深度学习入门笔记

第一章 Python入门

1. Python3.x写的代码不能被Python2.x所执行。
2. Python属于动态类型语言，变量类型根据情况自动决定。
3. Python的逻辑运算符 `and or not`
4. Python构造类的格式：

```
class className:
   def __init__(self,parameters):
      xxx
   def function1(self,parameters):
      xxx
   def function2(self,parameters):
      xxx
```

   `__init__` 方法是进行初始化，也称构造函数(constructor)只在生成类的实例时被调用一次。
5. *一个奇技淫巧* `X[X>15]` 取出方括号内值为 `True` 的元素。
6. ```
   plt.plot(x,y,linestyle='',label='')#参数分别表示横轴，纵轴，线形和标签
   plt.xlabel('') #横轴标签
   plt.ylabel('') #纵轴标签
   plt.title('') #标题
   plt.legend() #显示图例
   plt.show() #显示图像
   ```
7. matplotlib读取和显示图像
   ```

```
import matplotlib.pyplot as plt
from matplotlib.image import imread
img = imread('path.png')
plt.imshow(img)
plt.show()
```

第二章 感知机

1. 感知机(perceptron)接收多个信号，输出一个信号。
   输入信号被送往感知机（神经元）时会分别乘以固定的权重(w_1x_1、w_2x_2)神经元会计算信号的总和，超过阈值时输出1，称为神经元被激活，阈值用符号θ表示。
   公式表示为：

$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leqslant \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$

2. 感知机的行为：

$$y = \begin{cases} 0 & (b + w_1x_1 + w_2x_2 \leqslant 0) \\ 1 & (b + w_1x_1 + w_2x_2 > 0) \end{cases}$$

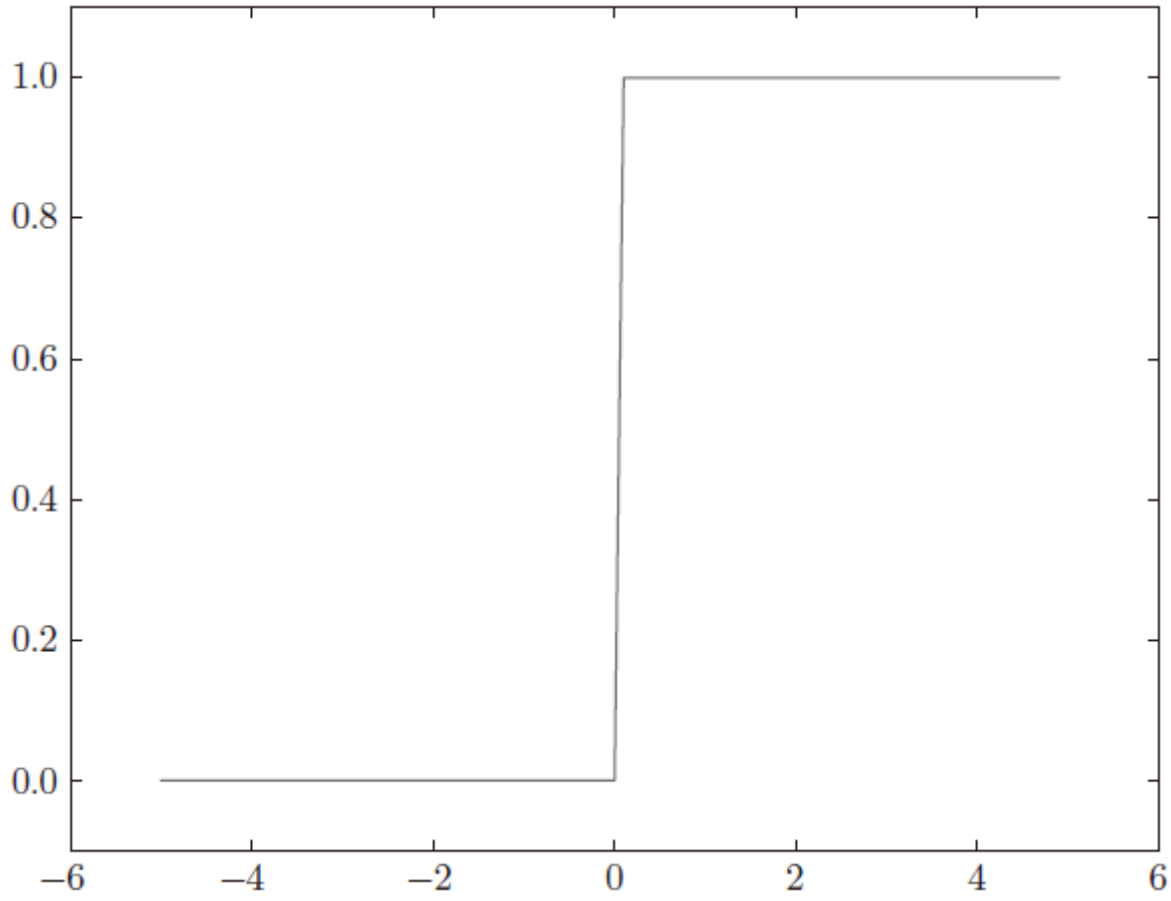
   b称为偏置，调整神经元被激活的容易程度，w_1和w_2称为权重，控制输入信号的重要性参数。

第三章 神经网络

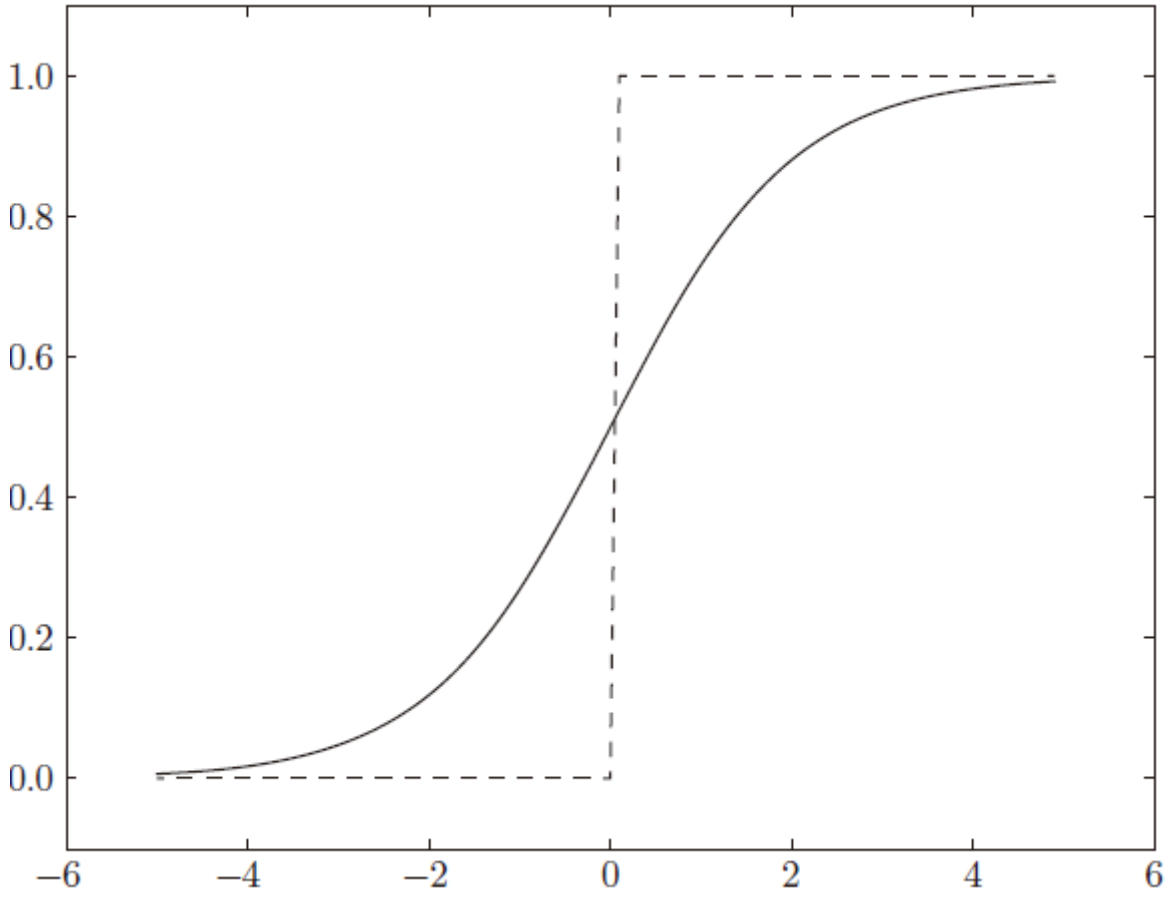
1. 用图表示神经网络，最左边的称为**输入层**，最右边一列称为**输出层**，中间的称为**中间层**，有时也称为**隐藏层**。
2. 将输入信号的总和转换为输出信号称为**激活函数**(activation function)。
   - sigmoid函数(sigmoid function)：

$$h(x) = \frac{1}{1 + \exp(-x)}$$

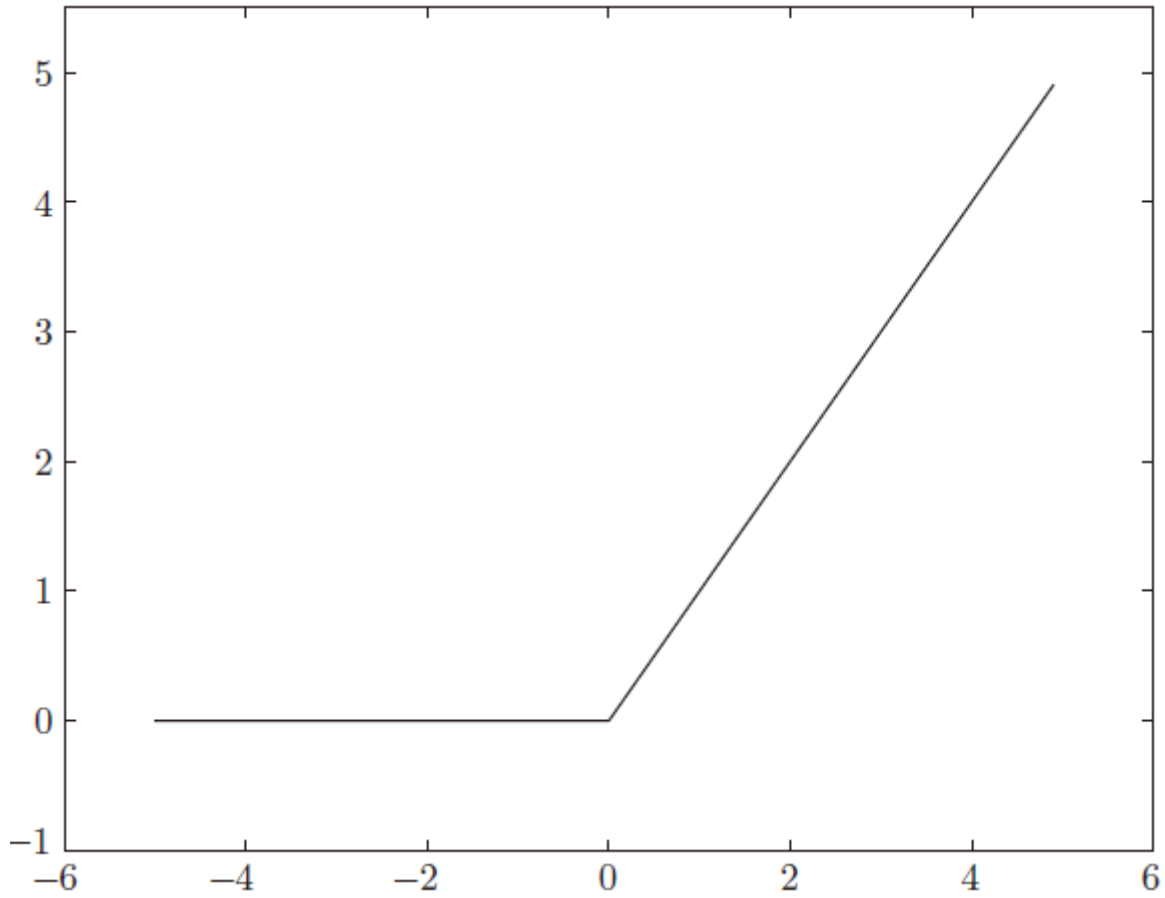
   - 阶跃函数图像



sigmoid函数与阶跃函数



- ReLU函数



3. 输出层的激活函数，回归问题用**恒等函数**，二元分类函数使用**sigmoid函数**，多元分类函数使用**softmax函数**。

   softmax函数：

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)}$$

   这一的函数在计算机中在数值很大的情况下可能出现溢出的情况，因此对其改造：

$$\begin{aligned} y_k = \frac{\exp(a_k)}{\sum_{i=1}^{n} \exp(a_i)} &= \frac{\mathrm{C}\exp(a_k)}{\mathrm{C}\sum_{i=1}^{n} \exp(a_i)} \\ &= \frac{\exp(a_k + \log \mathrm{C})}{\sum_{i=1}^{n} \exp(a_i + \log \mathrm{C})} \\ &= \frac{\exp(a_k + \mathrm{C}')}{\sum_{i=1}^{n} \exp(a_i + \mathrm{C}')} \end{aligned}$$

   softmax的输出为0.0-1.0之间的实数，可称之为概率。

4. 把数据限定到某个范围内的处理称为**正规化(normalization)**.对神经网络进行某种既定的转换称为**预处理(pre-processing)**。
5. 打包式的输入数据称为**批(batch)**.可大幅缩短处理时间。

第四章 神经网络的学习

1. 神经网络的学习是指从训练数据中自动获取最优权重参数的过程，引入损失函数，找到使其值达到最小的权重参数。
2. 一般将数据分为**训练数据**（也称为监督数据）和**测试数据 过拟合 泛化能力**
3. **损失函数**(loss function)一般用均方误差和交叉熵误差。
   - 均方误差(mean squared error)：

$$E = \frac{1}{2}\sum_k (y_k - t_k)^2$$

     其中y_k表示神经网络的输出，t_k表示监督数据，k表示数据的维数
   - 交叉熵误差(cross entropy error):

$$E = -\sum_k t_k \log y_k$$

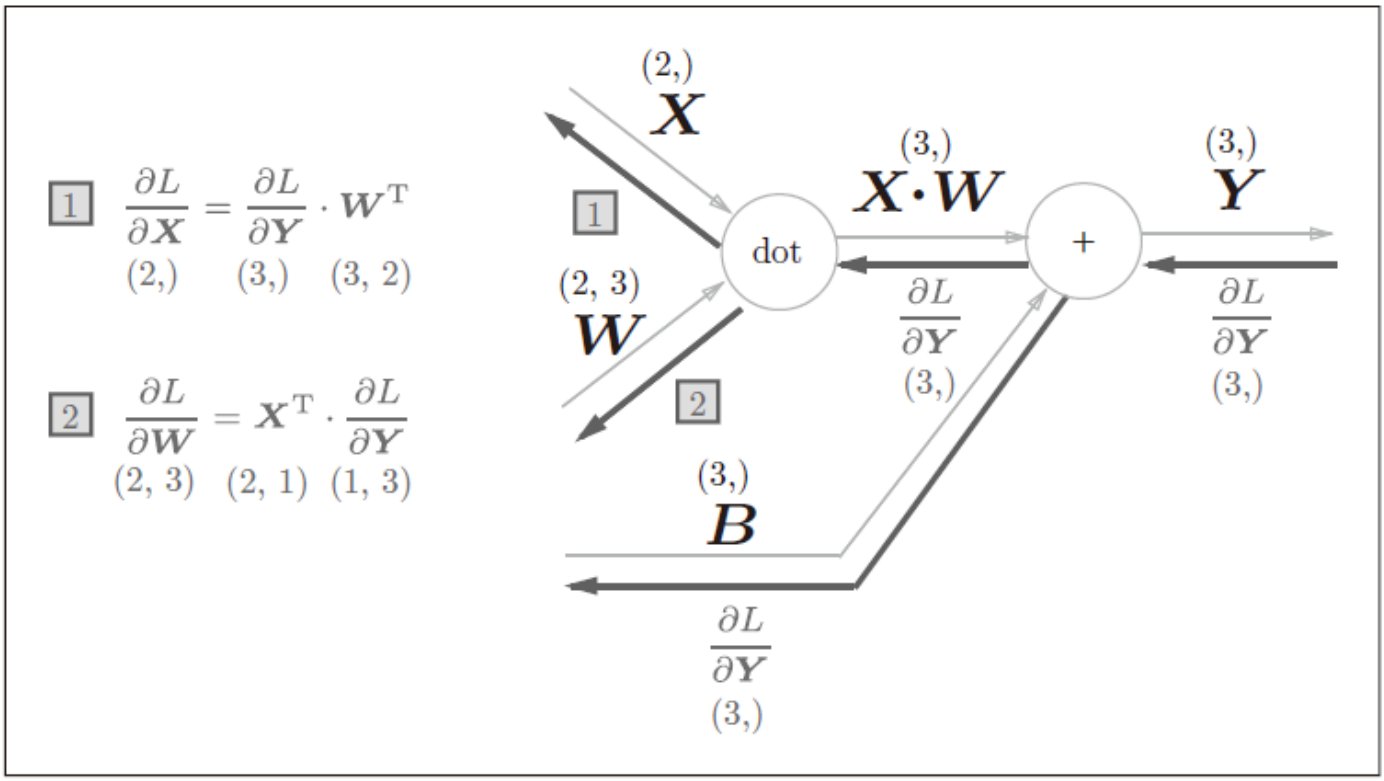
     正确解标签所对应的输出结果决定。
4. **mini-bath学习**（小批量学习）从训练数据中选出一批数据(mini-batch)，然后对每个mini-batch进行学习。
5. 引入损失函数的原因：不能以识别精度作为指标，参数的导数在绝大多数地方会变为0.
6. **中心差分**函数*f*在(*x*+*h*)和(*x*-*h*)之间的差分；前向差分(*x*+*h*)和*x*之间的差分。
7. **梯度**(gradient)由全部变量的偏导数汇总而成的向量。 梯度指示的方向是各点处函数值减小最多的方向。
8. **梯度法**从当前位置沿着梯度方向前进在新的地方重新求梯度，不断反复，逐渐减小函数值的过程。严格的讲寻找最小值的称为梯度下降法；寻找最大值的称为梯度上升法。
9. $$x_0 = x_0 - \eta\frac{\partial f}{\partial x_0}$$
$$x_1 = x_1 - \eta\frac{\partial f}{\partial x_1}$$

   其中η表示更新量，在神经网络中称为学习率(learning rate).像这样的参数称为超参数。
10. 随机梯度下降法(stochastic gradient decent)一般称为SGD。

11. epoch 表示学习中所有训练数据均被使用过一次时的更新次数。（将所有训练数据随机打乱，按指定的批次大小生成mini-batch，每个都有一个索引号，遍历一次就成为一个epoch）

第五章 误差反向传播法

1. **计算图** 数据结构图，通过多个节点和边表示（连接节点的直线称为边） 可以通过正向传播和反向传播高效的计算各个变量的导数值。
2. 加法节点的反向传播只乘以1，输入值原封不动的流向下一节点。 乘法的反向传播将上游的值诚意正向传播时的输入信号的翻转值，正向传播时的信号是x的话，反向传播则是y。
3. Affine层的反向传播；



批版本的Affine层传播：

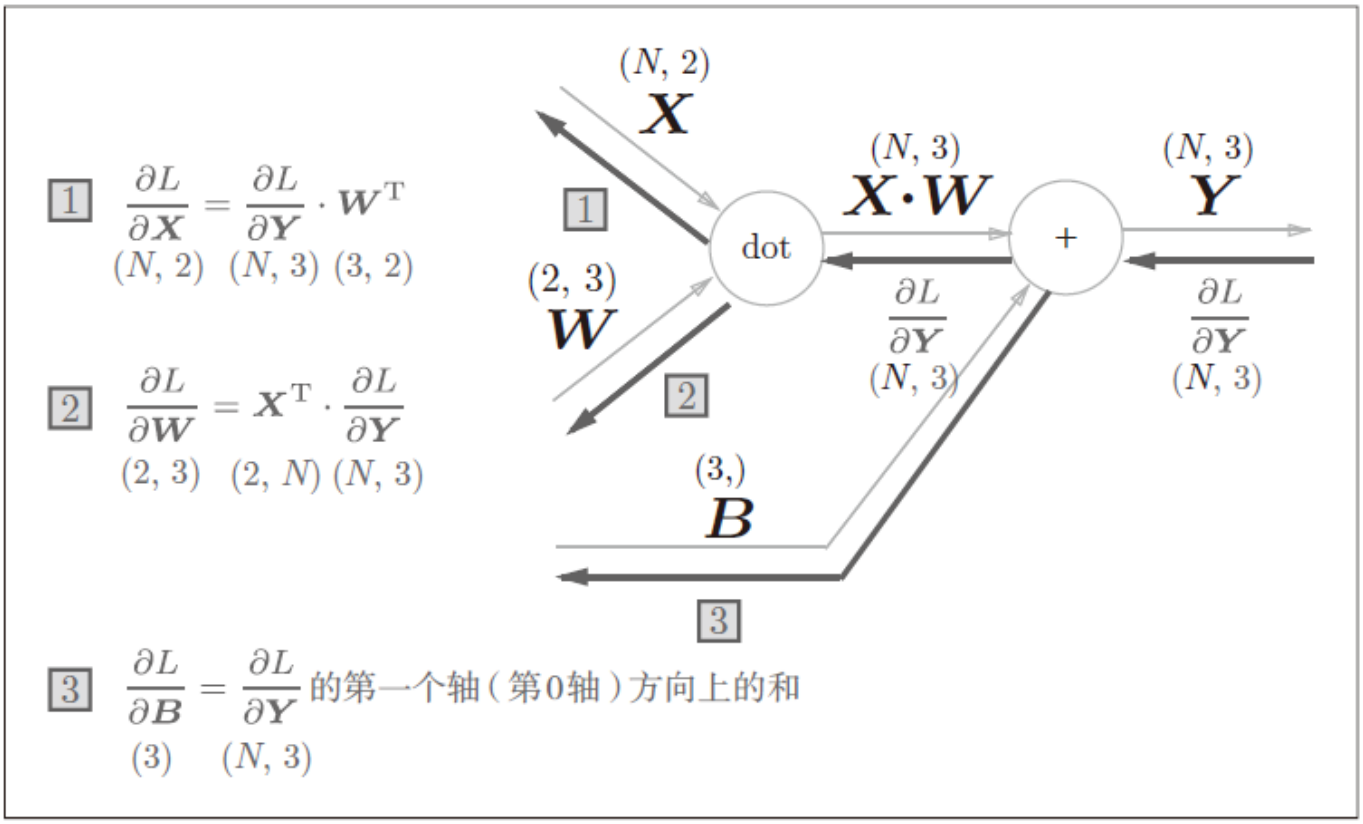


图5-27　批版本的Affine层的计算图

4. 神经网络进行的处理有推理和学习两个阶段，神经网络的推理通常不使用Softmax层，神经网络中未被正规化的输出结果被称为“得分”。神经网络的学习阶段需要Softmax层。
5. 神经网络学习的全过程
    - 前提 神经网络中有合适的权重和偏置，调整权重和偏置来拟合训练数据的过程称为学习；
    - 步骤1 mini-batch 从训练数据中随机选择一部分数据；
    - 步骤2 计算梯度 计算损失函数关于各个权重参数的梯度；（误差反向传播）
    - 步骤3 更新参数 将权重参数沿梯度方向进行微小的更新；
    - 步骤4 重复 重复步骤1、2、3.

第六章 与学习相关的技巧

1. 解决寻找最优参数的问题称为最优化(optimization)。
2. SGD的缺点是，如果函数的形状非均向，比如呈延申状，函数的路径会非常的低效，其根本原因是梯度的方向没有指向最小值的方向。
3. Momentum法

$$\boldsymbol{v} \leftarrow \alpha\boldsymbol{v} - \eta\frac{\partial L}{\partial \boldsymbol{W}}$$
$$\boldsymbol{W} \leftarrow \boldsymbol{W} + \boldsymbol{v}$$

*v*对应物理上速度，第一式表示了物体在梯度方向上受力。

4. AdaGrad 学习率衰减(learning rate decay)随着学习的进行，使学习率逐渐减小。

$$\boldsymbol{h} \leftarrow \boldsymbol{h} + \frac{\partial L}{\partial \boldsymbol{W}} \odot \frac{\partial L}{\partial \boldsymbol{W}}$$
$$\boldsymbol{W} \leftarrow \boldsymbol{W} - \eta \frac{1}{\sqrt{\boldsymbol{h}}} \frac{\partial L}{\partial \boldsymbol{W}}$$

新变量*h*保存了所有梯度值的平方和，式中运算表示矩阵元素的乘法。下式通过系数调整学习率，参数的元素中变动大的学习率将减小。

5. Adam 直观的讲就是融合了Momentum和AdaGrad的方法。
6. 很多研究人员和技术人员都喜欢用Adam.一般而言与SGD相比其他三种方法学习的更快，有时最终的识别精度也更高。
7. 抑制过拟合,提高泛化能力的技巧--权值衰减(weight decay)以减小权重参数的值为目的进行学习,来抑制过拟合的发生.
8. 权重的初始值不能设为相同的值,为了防止权重均一化(严格讲是为了瓦解权重的对称结构),必须随机生成初始值.
9. 偏向0和1的数据分布会造成反向传播中梯度的值不断变小,最后消失,称之为**梯度消失**(gradient vanishing).
10. 当激活函数使用ReLU时,权重初始值使用He初始值,当激活函数为sigmoid或tanh等S型曲线函数时,初始值使用Xavier初始值.这是目前嘴贱的实践. **在神经网络学习中,权重初始值非常重要,其设定关系到神经网络的学习能否成功.**
11. Batch Normalization(2015)
    - 可以使学习快速进行(可以增大学习率)
    - 不那么依赖初始值(对于初始值不用那么神经质)
    - 抑制过拟合(降低Dropout等的必要性)

    其思路是调整各层的激活值分布使其拥有适当的广度,因此要向神经网络中插入对数据分布进行正规化的层,即Batch Normalization层.如:

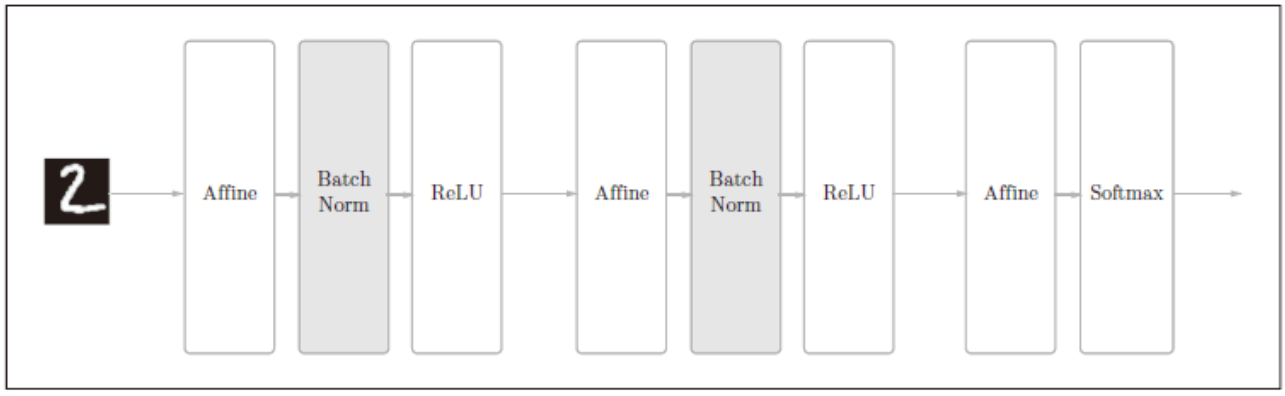


图6-16　使用了Batch Normalization的神经网络的例子(Batch Norm层的背景为灰色)

顾名思义,按mini-batch进行正规化,具体而言就说进行使数据分布的均值为0,方差为1的正规化,数学表达式为:

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

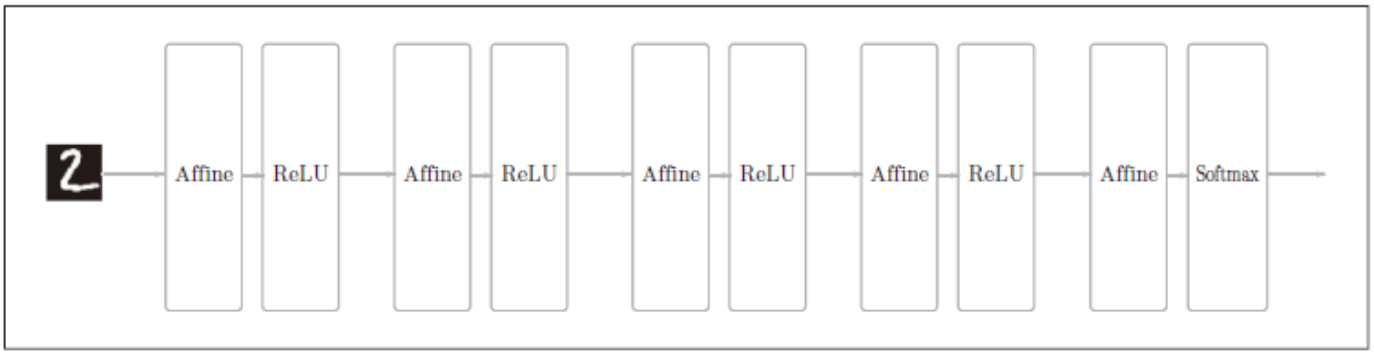
这里对*m*个输入数据的集合B求均值和方差,然后对数据进行均值为0,方差为1的正规化.将这个处理插入到激活函数的前面(或后面),可以减小数据分布的偏向.**综上**,通过使用Batch Normalization可以推动学习的进行,并且,对权重初始值变得健壮(表示不那么依赖初始值).

第六章 与学习相关的技巧

1. **正则化 过拟合**指的是只能拟合训练数据,但不能很好的拟合不包含在训练数据中的其他数据的状态.机器学习的目标是提高泛化能力,即便是不包含在训练数据中的未观测数据也希望模型正确的识别.发生过拟合的原因:
   - 模型拥有大量的参数,表现力强;
   - 训练数据少.
2. **权值衰减** 经常被使用的一种抑制过拟合的方法,通过在学习过程中对大的权重进行惩罚来抑制过拟合. 为损失函数加上权重的平方范数(L2范数)
3. **Dropout** 网络模型很复杂的情况下使用 是一种再学习的过程中随机删除神经元的方法.训练时随机选出隐藏层的神经元,然后将其删除,被删除的神经元不再进行信号的传递.测试时，虽然会传递所有神经元信号，但对于各个神经元的输出，要乘上训练时的删除比例后再输出。
   在简单的实现中，正向传播时传递了信号的神经元，反向传播时按原样传递信号；正向传播时没有传递信号的神经元，反向传播时信号将停在那里。
4. 集成学习 让多个模型单独进行学习，推理时再取多个模型的输出的平均值，通过集成学习，神经网络的识别精度可以提高好几个百分点，Dropout将集成学习的效果（模拟的）通过一个网络实现。
5. 超参数(hyper-parameter)比如各层的神经元数量、batch大小、参数更新时的学习率或权值衰减等。
6. 不能使用测试数据评估超参数的性能，这一点非常重要。如果使用测试数据调整超参数，超参数的值会对测试数据发生过拟合，换言之，用测试数据确认超参数值的好坏，会导致超参数的值被调整为只拟合测试数据，可能得到不能拟合其他数据、泛化能力低的模型。

7. 调整超参数是，必须使用超参数专用的确认数据，用于调整超参数的数据，一般称为**验证数据**(validation data)，用验证数据来评估超参数的好坏。
8. **训练数据**用于参数的学习，**验证数据**用于超参数的性能评估，为了确认泛化能力，要在最后使用测试数据。
9. 超参数的优化：
    - **步骤0** 设定超参数的范围
    - **步骤1** 从设定的超参数范围中随机采样
    - **步骤2** 使用步骤1中采样到的超参数的值进行学习，通过验证数据评估识别精度(但是要将epoch设置的很小)
    - **步骤3** 重复步骤1和步骤2(100次等)，根据他们的识别精度的结果缩小超参数的范围。缩小到一定的程度时，从该范围中选出一个超参数的值。
      在超参数的最优化中，如果需要更严密、高效的进行优化，可以使用**贝叶斯最优化**(Bayesian optimization)

第七章 卷积神经网络

1. 卷积神经网络(Convolutional Neural Network,CNN) 出现了新的卷积层(Convolution层)和池化层(Pooling层)。
2. 基于全连接层(Affine层)的网络的例子
3. 

基于CNN的网络例子

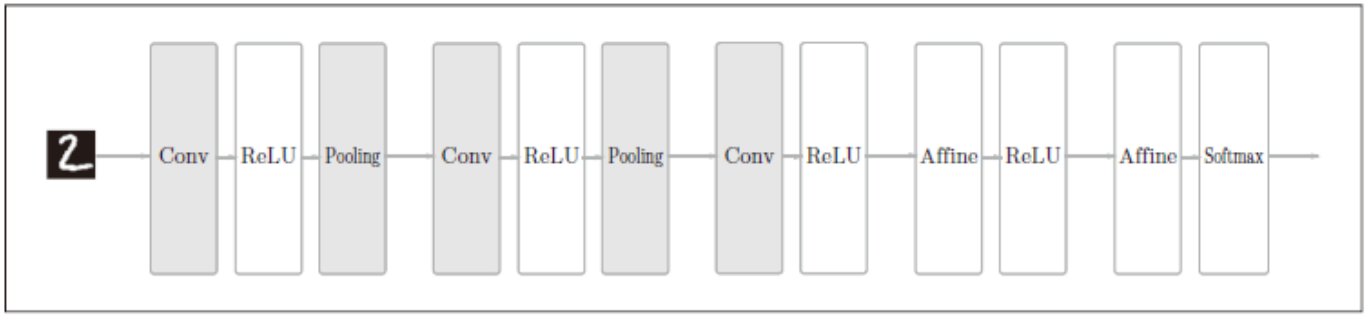


图7-2　基于CNN的网络的例子：新增了Convolution层和Pooling层(用灰色的方块表示)

CNN的层的连接顺序是"Convolution-ReLU-(Pooling)"，靠近输出的层使用了之前的"Affine-ReLU"组合，最后的输出层中使用了之前的"Affine-Softmax"组合，这些都是一般的CNN中比较常见

的结构。

4. 全连接层存在的问题是数据的形状被忽略了，有时将卷积层的输入输出数据称为特征图(feature map)，其中卷积层的输入数据称为输入特征图(input feature map)，输出数据称为输出特征图(output feature map)。
5. 卷积层进行的处理就是卷积运算，相当于图像处理中的滤波器运算，有文献中也使用“核”来表示滤波器。 CNN中滤波器的参数就对应之前的权重，CNN中也存在偏置。
6. **填充** 在进行卷积层的处理之前有时要向输入数据的周围填入固定的数据,这称为填充(padding)
7. **步幅** 应用滤波器的位置间隔称为步幅(stride)
8. 综上,增大步幅后输出大小会变小,增大填充后输出大小会变大.假设输入大小为(*H,W*),滤波器的大小为(*FH,FW*),输出大小为(*OH,OW*),填充为*P*,步幅为*S*,此时输出大小可表示为:

$$OH = \frac{H + 2P - FH}{S} + 1$$
$$OW = \frac{W + 2P - FW}{S} + 1$$

9. 三维数据的卷积运算 通道方向上有多个特征图时,会按通道进行输入数据和滤波器的卷积运算,并将结果相加从而得出输出.
10. 三维数据表示为多维数组时,书写顺序为(channel,height,width),滤波器也一样.

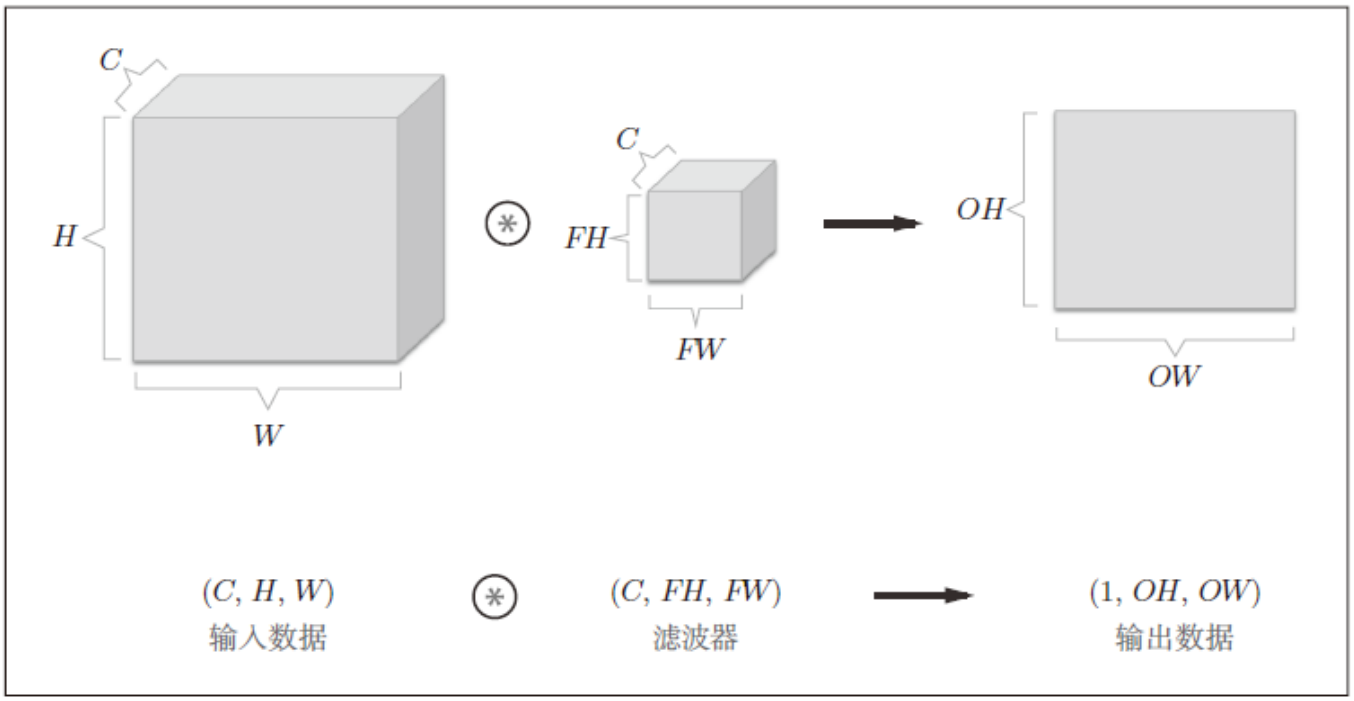


图7-10　结合方块思考卷积运算。请注意方块的形状

上例中数据输出的是1张特征图,就是通道数为1的特征图

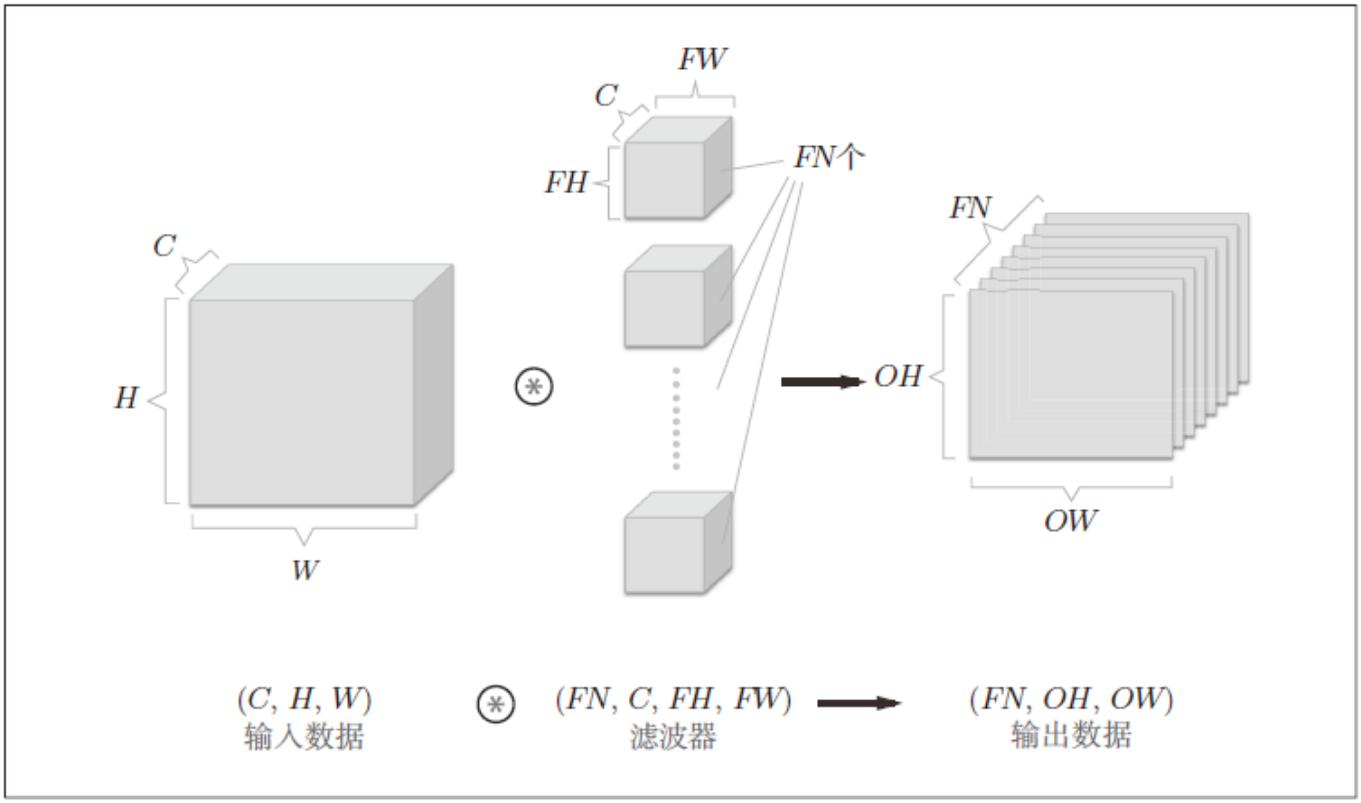


图7-11　基于多个滤波器的卷积运算的例子

通过应用FN个滤波器,输出特征图也生成了FN个.

11. 滤波器的权重数据要按(output_channel,input_channel,height,width)的顺序书写.
12. CNN中的批处理 按(batch_num,channel,height,width)的顺序保存数据.批处理将N次处理汇总成了1次进行

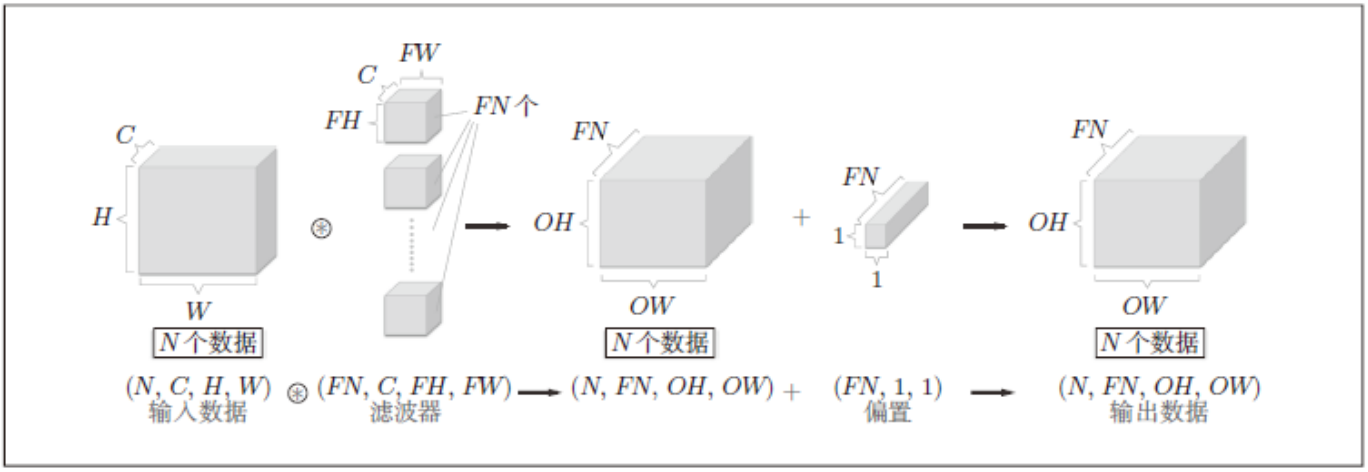


图7-13　卷积运算的处理流(批处理)

13. 池化层 池化是缩小高,长方向上的空间的运算,如将2x2的区域集约成一个元素的处理,缩小空间的大小. 一般来说池化的窗口大小会和步幅设定成相同的值. 池化层有Max池化,Average池化等.图像识别领域主要使用Max池化.
14. 池化层有以下特征:
    - 没有要学习的参数 只是从目标区域中取最大值(或者平均值),所以不存在要学习的参数.

- 通道数不发生变化
- 对微小位置变化具有鲁棒性(健壮) 输入数据发生微小偏差时,池化仍会返回相同的结果.

15. im2col 名称是image to column 从图像到矩阵,很多深度学习框架中都有这个函数,并在卷积层的实现中都使用了im2col.

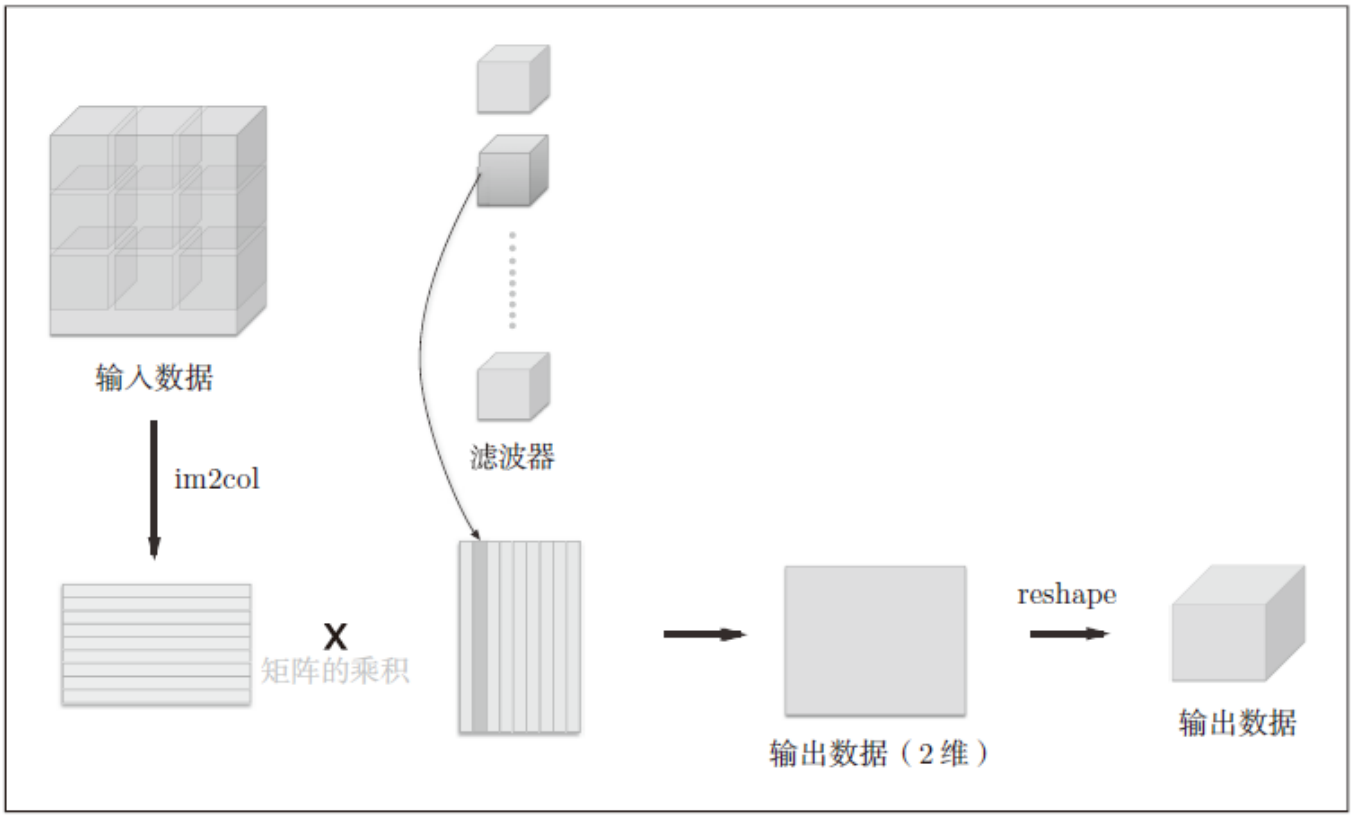


图7-19 卷积运算的滤波器处理的细节：将滤波器纵向展开为1列，并计算和im2col展开的数据的矩阵乘积，最后转换（reshape）为输出数据的大小

16. 卷积层的滤波器会提取边缘或斑块等原始信息,CNN会将这些原始信息传递给后面的层. 如果堆叠了多层卷积层,则随着层次的加深,提取的信息也愈加复杂抽象.
17. 具有代表性的CNN:
    - LeNet(1998) 手写数字识别的网络 有连续的卷积层和池化层,最后经全连接层输出结果.LeNet使用sigmoid函数,而现在的CNN主要使用ReLU函数,原始的LeNet中使用子采样(subsampling)缩小中间数据的大小,而现在的CNN中Max池化是主流.
    - AlexNet(2012) AlexNet叠有多个卷积层和池化层,最后经由全连接层输出结果,激活函数使用ReLU,使用进行局部正规化的LRN(Local Response Normalization)层,使用Dropout.

第八章 深度学习

1. 深度学习是加深了层的深度神经网络.
2. 有助于提高识别精度的方法:集成学习,学习率衰减,Data Augmentation(数据扩充).尤其是数据扩充虽然方法简单,但在提高识别精度上效果显著.

3. 数据扩充基于算法人为的扩充输入图像(训练图像),具体的说,对于输入图像通过施加旋转,垂直或水平方向上的移动等微小变化,增肌图像的数量,这在数据集的图像数量有限时尤为有用.此外还有剪裁图像的crop处理,左右翻转图像的flip处理,施加亮度,放大缩小尺度变化.
4. 加深网络层的好处:减少网络的参数数量;叠加小型滤波器来加深网络的好处是可以减少参数的数量,扩大感受野(receptive field,给神经元施加变化的某个局部空间区域);另一个好处是使学习更加高效,通过加深层,可以将各层要学习的问题分解成容易解决的简单问题.
5. VGG是由卷积层和池化层构成的基础的CNN,其特点在于将有权重的层(卷积层或者全连接层)叠加至16层(或者19层)具备了深度(根据层的深度,有时也成为VGG16或VGG19),需要注意的地方是基于3x3的小型滤波器的卷积层的运算是连续进行的.
6. GoogLeNet 网络不仅在纵向上有深度在横向上也有深度(广度)
7. ResNet 具有比以前的网络更深的结构,引入快捷结构(捷径)解决网络加深导致学习不能顺利进行的问题,即使加深层也能高效的学习,通过快捷结构反向传播时信号可以无衰减地传递,梯度消失问题有望得到缓解.
8. **迁移学习** 将学习完地权重(一部分)复制到其他神经网络,进行再学习(fine tuning).比如,准备一个和VGG相同结构地网络,把学习完的权重作为初始值,以新数据集为对象进行再学习,迁移学习在手头数据集较少时非常有效.
9. GPU可以高速地处理大量地运算,最近地框架也开始支持多个GPU或多台机器上地分布式学习.
10. GPU计算是指基于GPU进行通用地数值计算地操作.
11. 为了实现深度学习的高速化，位数缩减是今后必须关注的一个课题，特别是在面向**嵌入式应用程序**中使用深度学习时，位数缩减非常重要。
12. 深度学习的部分案例：
    - 物体检测 从图像中确定物体的位置并进行分类 使用R-CNN 使用流程：1、Input image;2、Extract region proposals(候选区域特征提取);3、Compute CNN features(CNN特征计算);4、Classify regions.R-CNN会将图像变形为正方形,或者在分类时使用SVM.
    - 图像分割 在像素水平上对图像进行分类
    - 图像标题的生成 NIC(Neural Image Caption)模型 由深层的CNN和处理自然语言的RNN(Recurrent Neural Network)构成,RNN是呈递归式连接的网络，经常被用于自然语言,时间序列数据等连续性的数据上.NIC基于CNN从图像中特区特征，并将这个特征传给RNN，RNN以CNN提取出的特征为初始值，递归的生成文本.(Recurrent递归的 是指神经网络的递归的网络结构,神经网络会受到之前生成的信息的影响)
13. **多模态处理** 将组合图像和自然语言等多种信息进行的处理称为多模态处理.
14. 深度学习的未来:
    - 图像风格转换 输入两个图像后,会生成一个新的图像,两个输入图像中,一个称为"内容图像",一个称为"风格图像".
    - 图像的生成 例如:基于深度学习,可以实现从零生成卧室的图像.(GAN)
    - 自动驾驶　路线规划(path plan)技术,相机激光雷达传感技术.基于CNN的神经网络SegNet可以高精度地识别行使环境.
    - Deep Q-Network 就像人类通过摸索试验来学习一样,让计算机也在摸索试验地过程中自主学习,称为**强化学习**(reinforcement learning),强化学习的基本框架是,Agent根据环境选择行动然后通

过这个行动改变环境,根据环境的改变,Agent获得某种报酬,强化学习的目的是决定Agent的行动方针,以获得更好的报酬.DQN基于被称为Q学习的强化学习算法,在Q学习中,为了确定最合适的行动,需要确定一个被称为最优行动价值函数的函数,为了近似这个函数,DQN使用了深度学习.