lec14

反向传播算法的基本概念和计算过程，特别是单个神经元的梯度计算，以及卷积神经网络（ConvNets）的基本原理和应用，包括卷积和池化操作的技术细节

一个神经元的反向传播算法

- 误差函数：假设误差函数 $E=\frac{1}{2}(o-d)^2$，其中 o 是输出，d 是期望输出
- 激活函数：使用 sigmoid 函数 f 作为激活函数，那么 $o=f\left(\sum_{j=0}^{n} w_j x_j\right)$
- 表达式分解：将误差函数 E 分解为不同部分，便于计算梯度
- 然后我们找到各部分的（偏）导数：

$$E=\frac{1}{2}(o-d)^2 \quad \text{其中} \quad o=f(q) \quad \text{且} \quad q=\sum_{j=0}^{n} w_j x_j$$

- 这给我们：

$$\frac{\partial E}{\partial o}=o-d, \quad \frac{\partial o}{\partial q}=f(q)(1-f(q)), \quad \text{且} \quad \frac{\partial q}{\partial w_i}=x_i$$

- 然后，通过链式法则：

$$\frac{\partial E}{\partial w_i}=\frac{\partial E}{\partial o}\cdot\frac{\partial o}{\partial q}\cdot\frac{\partial q}{\partial w_i}=(o-d)(o(1-o))x_i$$

  其中

$$o=\frac{1}{1+e^{-w\cdot x}} \quad \text{且} \quad w\cdot x=\sum_{j=0}^{n} w_j x_j$$

- 因此我们知道误差对权重的梯度：

$$\frac{\partial E}{\partial w_i}=(o-d)(o(1-o))x_i$$

  其中

$$o=\frac{1}{1+e^{-w\cdot x}} \quad \text{且} \quad w\cdot x=\sum_{j=0}^{n} w_j x_j$$

- 然后，我们希望根据梯度更新权重，以找到误差函数的最小值：

$$\Delta w_j=-\eta(o-d)(o(1-o))x_i$$

- 最后

$$w_j \leftarrow w_j+\Delta w_j \quad \text{对于所有} \quad j=0,\ldots,n$$

ALGORITHM Backpropagation

Start with randomly chosen weights

WHILE MSE is above desired threshold and computational bounds are not exceeded, **DO**

FOR EACH input pattern x_p, $1 \le p \le P$,

Compute hidden node inputs

Compute hidden node outputs

Compute inputs to the output nodes

Compute the network outputs

Compute the error between output and desired output

Modify the weights between hidden and output nodes

Modify the weights between input and hidden nodes

END-FOR

END-WHILE

9

卷积和滤波

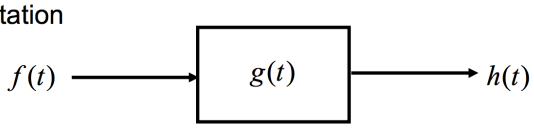
- 噪声去除 Noise removal
  - 通过滤波将噪声从信号中分离出来
  - 周期性噪声去除
- 从图像中提取信息
  - 边缘、特征点等（Edge detection）
- 模式检测
  - 模板匹配（template matching）
- 反卷积
  - 去除先前应用的线性操作的影响（remove effects of previously applied linear operations）

1D卷积

连续卷积

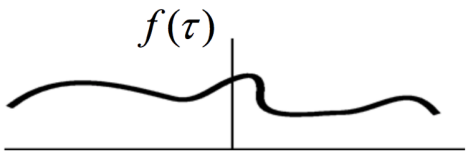
- 系统组成：
  - 输入信号 $f(t)$
  - 传递函数 $g(t)$
  - 输出信号 $h(t)$
- 卷积定义：
  - 如果一个信号 $f(t)$ 输入到一个线性系统中，输出 $h(t)$ 是输入信号和传递函数 $g(t)$ 的卷积

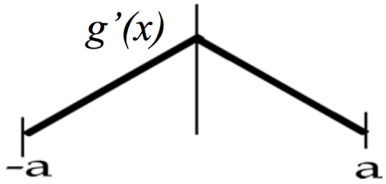
Notation

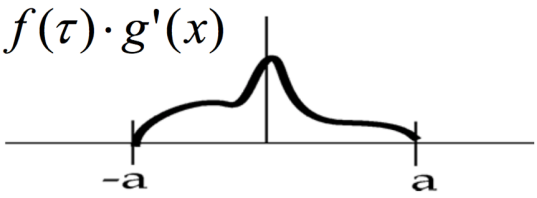


$$h(t) = \int_{-\infty}^{+\infty} g(t-\tau) f(\tau) d\tau$$

- 通过卷积进行滤波：
  - 计算乘积曲线下面的面积 $h(x) = f(\tau) \cdot g'(x)$



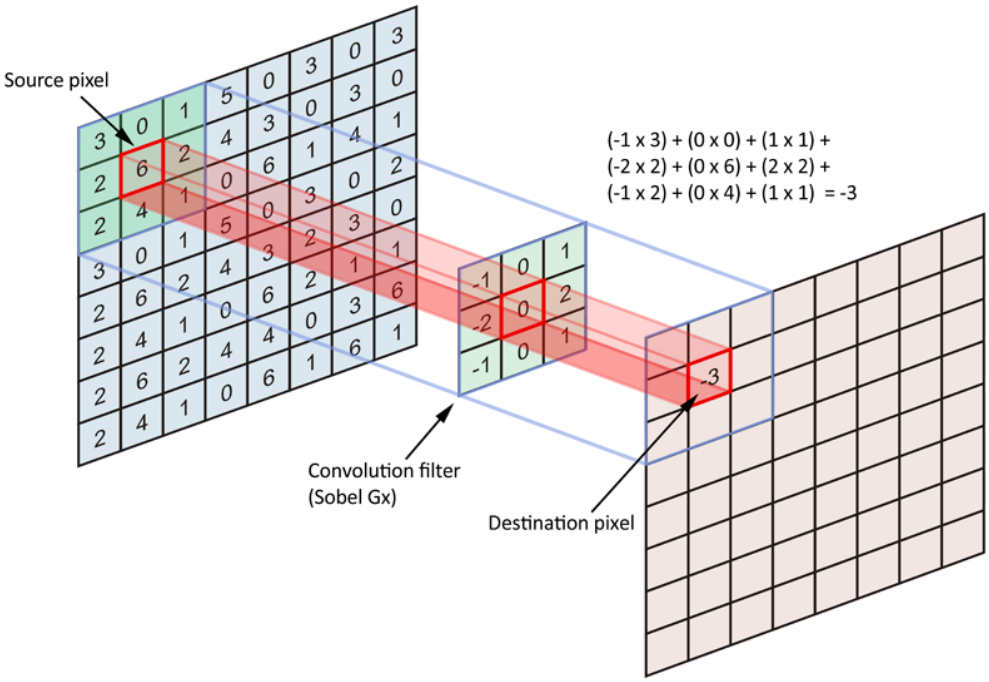


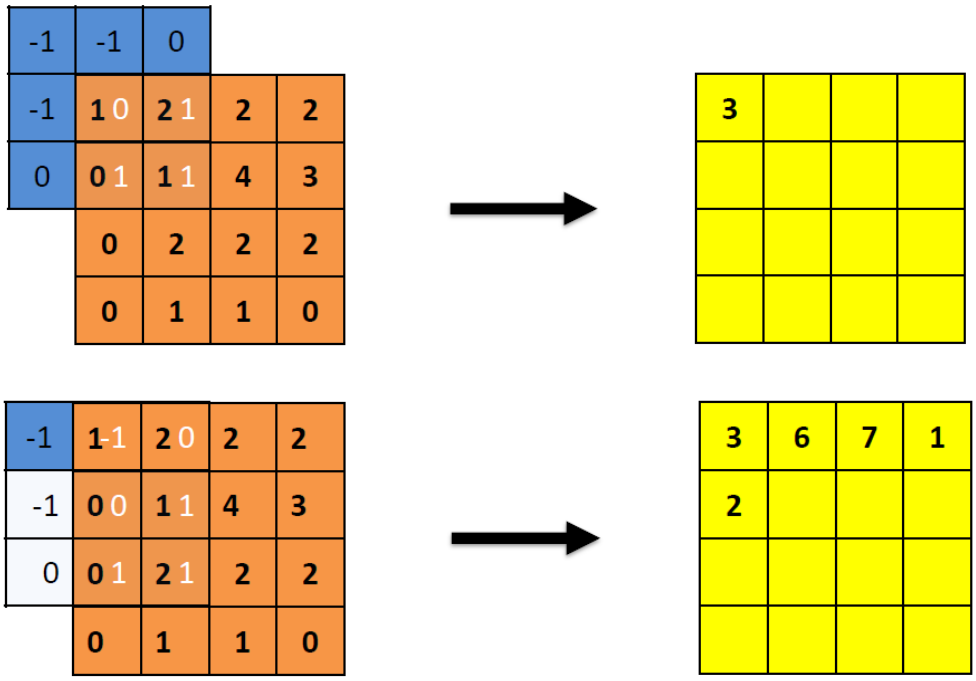


离散卷积

- 输入:
  - 具有m 个采样点的离散信号 f
- 卷积核:
  - 长度为 n 的滤波器 g
- 输出:
  - 长度为 m 的序列 h
- 第 i 个元素:
  - $h(i) = f(i) \times g(i) = \sum f(j)g(i-j)$

2D卷积





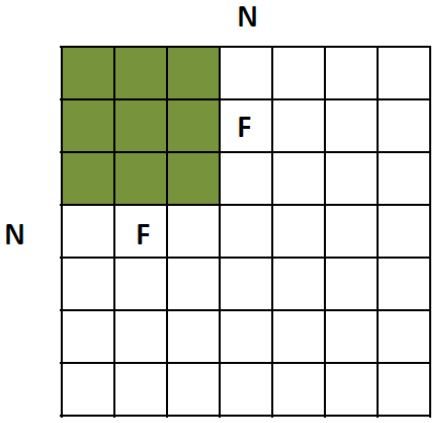
stride

Stride（步幅）是卷积运算中的一个重要参数，决定了卷积核在输入矩阵上滑动的步长。Stride的大小影响输出矩阵的大小和卷积运算的计算效率。通过调整Stride，可以在计算复杂度和输出精度之间找到平衡

1. Stride为1：
   - 当Stride为1时，卷积核在输入矩阵上每次滑动一个像素
   - 这种情况下，输出矩阵的大小较大，保留了更多的输入信息
2. Stride为2：
   - 当Stride为2时，卷积核在输入矩阵上每次滑动两个像素
   - 这种情况下，输出矩阵的大小较小，计算量减少，但可能丢失一些细节信息

输出大小：

$(N - F)/stride + 1$



Output size:
(N - F) / stride +1

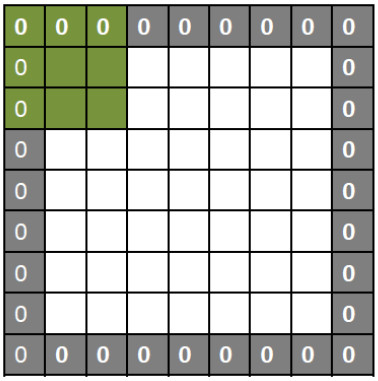
e.g., N=7, F=3:
stride 1 => (7 - 3) / 1 + 1 = 5
stride 2 => (7 - 3) / 2 + 1 = 3
stride 3 => (7 - 3) / 3 + 1 = 2.33

padding

Padding（填充）是卷积操作中的一个重要参数，用于控制输出矩阵的大小。通过在输入矩阵的边界添加额外的像素，可以保持输入和输出矩阵的尺寸一致，或者避免边界效应。常用的填充方式有无填充和零填充。零填充通过在输入矩阵的边界添加零值像素，扩展输入矩阵的尺寸，从而影响输出矩阵的大小

padding大小：

$\frac{F-1}{2}$



In practice, it is common to zero pad the border

e.g., input 7x7
3x3 filter, applied with stride 1
pad with 1 pixel border => what is the output?

7x7 output!

In general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g., F=3 => zero pad with 1
F=5 => zero pad with 2
F=7 => zero pad with 3