

2022-03-20

计算机视觉导论 (北京大学 2022 春)

第一次课程作业笔记 by PkuCuipy

⚠ 请独立完成课程作业!

报告及代码仅供参考, 欢迎 issue 讨论代码中的问题和改进办法, 提前表示感谢!

另附我的完成时间如下, 供任务量参考:

第一题: 2.5h; 第二题: 3.5h; 第三题: 0.5h; 第四题: 3h; 第五题: 2h

1. Convolution operation (15')

1(a) Implementing Padding Function

- 在 Replicate Padding 中, 要注意四个角上的复制的规则.

1(b) Implementing 2D Convolution via Toeplitz Matrix

- Toeplitz Matrix 的构造参考 [2-D convolution as a matrix-matrix multiplication](#).
- 题目暗示要用 fancy indexing 构造 Toeplitz Matrix, 我不太能想得到, 而是用的 `np.roll()` 来实现. 因为矩阵 T 正是这种「每行是上一行的错开」的那种形状 (不完全是, 请自行推导 or 参考链接).

1(c) Convolution by Sliding Window

- 这个题要求用纯粹的不带 for 的 Numpy 来完成. 用到的索引方式我之前确实没用过, 是这样的: 对一个二维矩阵 A 而言, `A[[1,2,3]]` 意味着取出 A 的第 1、2、3 行, 然后构成一个矩阵; 而如果 `A[:,]` 中的 `:` 是二维矩阵, 则会得到一个三维矩阵, 每个子二维矩阵是用 `:` 的某一行对 A 进行索引得到的二维矩阵.
- 具体代码参考: [2D Image Convolution with Numpy with a Handmade Sliding Window View](#). 其思路是: 首先取出 012 行、123 行、234 行..., 然后对每个 [xyz] 行都再取 012 列、123 列、234 列...

2. Canny Edge Detector (20')

2(a) Compute the Image Gradient

- 由于涉及除法, 最好应该在分母上加上 Epsilon 防止出现 NaN. 我这里取的 `EPS = 1e-10`.

2(b) Non-Maximal Suppression (NMS)

- NMS 的思路就是: 对于 (m, n) 位置, 计算当前的值 A, 以及梯度正方向和梯度反方向走 1 格的值 B、C, 如果发现 $A > B$ 且 $A > C$, 则说明 A 是一个 Maximal, 否则就是 Non-Maximal, 需要被 Suppress.
- 这里由于 B、C 点不一定是网格点, 因此需要通过插值得到. 这里使用双线性插值 (BiLinear Interpolation).
- 并行化的双线性插值参考在 [numpy和python中简单有效的双线性插值图像](#). 要注意仔细判断链接里代码的 x 和 y 的方向是否和你理解的一致! (我就被这里坑到了, 见下)

- 用 Lenna 图 NMS 后的梯度有点断断续续的, 和 PPT 上的不太一样, 不知道是否正常?..
不正常! 最后发现把 grad_x 和 grad_y 交换一下顺序才对.
 这是因为: 题目理解的 x 和 y 方向是 x 向右, y 向下; 而我则想当然地认为 x 向下, y 向右. 真是\$%#!&了.

2(c) Edge Linking with Hysteresis Threshold

- 这里 PPT 上没有详细的细节, 于是参考了[这篇文章](#)和[这篇博客](#).

3. Harris Corner Detector (10')

- Slides 上讲得比较模糊, 我参考了知乎回答 [Harris角点响应函数表达式是怎么得来的?](#) .

4. Plane Fitting using RANSAC (25')

- RANSAC 算法参考[这个页面](#), 简短直观. 也参考了wiki 的[这个部分](#).
- 直觉上说, 就是现在有一堆点, 假设是 M 个. 但其中只有一部分能比较近似地拟合一条直线, 假设占比为 p , 称为 inlier; 另外的则比较离谱地分散各处, 后者被称之为 outlier. 此时如果使用最小二乘, 拟合的直线会被 outlier 严重影响!!
- RANSAC 的解决办法是:
 - 从这 M 个点中随机抽取 N 个点, 用这 N 个点做最小二乘拟合出一个直线 l. 然后检查 l 在 M 上的 inlier 个数 K.
 - 如果 K 目前最大, 就记录下直线 l.
 - 重复这样抽 L 次. L 可以预先计算, 使得在多少多少概率下 (比如 > 99.9%) 最好的 K 对应的 l 恰是包含所有 inlier 的.
- 对于这道题的数据而言, 130 个点, 其中 100 个点都完美地落在一个二维平面, 另外30个则散布在空间各处.
 要求预先给定轮数 L 以保证至少 99.9% 的概率最后能成功找到完美的平面.
- 计算 L:
 对于每一轮, 由于拟合平面只需要抽 3 个点, 那这三个点都是 inlier (从而成功) 的概率就是 $\frac{100}{130} \frac{99}{129} \frac{98}{128} \approx 0.452$.
 那么 L 轮至少成功一次的概率 $P_L = 1 - (1 - 0.452)^L$,
 令之 > 0.999, 得到 $(1 - 0.452)^L < 0.001$, 即 $L \ln(1 - 0.452) < \ln(0.001)$, 即 $L > \ln(0.001) / \ln(0.548) \approx 11.48$.
 那取 L = 12 就满足成功概率的要求.
- 对于每一组三个点, 都要对全部点计算是否为 inlier. 这显然是一个二重循环.
 如何使用 numpy 并行呢? 这里用到 CS231n HW1 的同款技巧, 那就是:
 (以这道题 groups.shape=(12, 3)、 points.shape=(130, 3) 为例)
 分别先将二者 reshape 到 (12, 1, 3) 和 (1, 130, 3),
 那么当这两个矩阵, 比如, 相加时, Numpy 就会自动将长度是 1 的维度 broadcast 到对方的相应尺寸!
 于是最终得到的结果就是 shape=(12, 130, 3) 的.
- 这里随机抽样要求并行地进行, 但我遇到了一些小问题, 注释里有提到.
 大概就是, 因为要抽取很多组点, 每组里面的三个点得是互异的, 因此要不放回抽样, 但组之间又理应是放回抽样的.
 所以我也不知道怎么一次性调用 Numpy 生成这样的随机数. 我在用了一个取巧的办法, 对于这道题倒是能完美解决.
 以及也是想吐槽, 这次作业要求是除非显式允许, 否则默认禁止使用 for 循环. 但对于这种其实没什么计算量的地方就很烦, 因为如果允许 for 循环, 这个地方就根本不是问题!

5. Backpropagation for an MLP (30')

- 做到这里, 忽然意识到这次作业的分值和难度似乎是成反比的. 这道题的难度绝对不值 30 分!

但还是 Debug 了快 1h, 最后发现是因为 sigmoid 那里求导搞错了! (印象里之前就在这个地方出过错..)

即: 如果令 $a(z) = \text{sigmoid}(z)$, 则应该有 $a'_z = a(1 - a)$, 而不是 $a'_z = z(1 - z)$!

- 另外就是关于: 对于 A、B、C 三个矩阵和相应的运算 $C = AB$, 什么叫 $\partial C / \partial A$?

我的理解是, 光这么看肯定是未定义的, 但如果最后计算出的 Loss 是一个数, 比如 `Loss = np.sum(C)`, 那么如果想问 $\partial L / \partial A$, 那么其实就可以拆成 $\partial L / \partial A = \partial L / \partial C \cdot \partial C / \partial A$.

观察等号右边, 第一项是有定义的, 因为是一个数去偏一个矩阵, 那就是个矩阵;

第二项就是刚才我们认为未定义的东西. 但这里**可以证明**这里缺失的东西其实是 B^T ,

所以我们就**形式化地**写成 $\partial C / \partial A = B^T$.

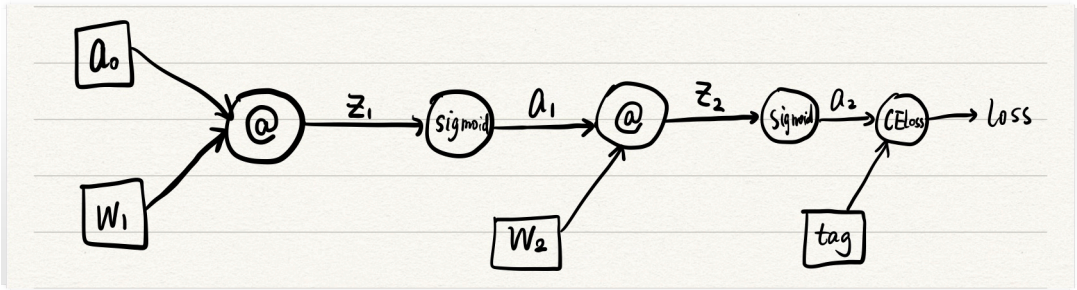
- 好吧, 上一段是我意淫的, 主要是参考的 [Backpropagation for a Linear Layer from Justin Johnson](#), 他这里是用了一个具体的例子来"证明"**这一点** (即上一段那个加粗的"可以证明"),

然后直接告诉你对于一般的情形也成立 (*but it holds for any values of N, D, and M*), 你爱信不信.

- 不过我的看法就是, 既然是线性的, 你考虑三个实数的运算 $c = ab$, 此时导数是很显然的 ($dc/da = b$), 而换成矩阵的话, 就多加一个转置即可.

如果记不住要不要转置, 技巧就是, 对于这种全是线性的运算, 看维数能否使得矩阵乘法合法即可!

- 本题代码中计算图部分涉及的变量名如下图所示:



第五题的计算图