计算机视觉作业

干皓丞，2101212850, 信息工程学院

2021 年 10 月 17 日

1 题目

Pytorch 程式码如下

```
import torch
torch.manual_seed(0)
x = torch.randn(10,4, requires_grad=True)
W = torch.randn(4,4, requires_grad=True)
y = torch.randn(10,4, requires_grad=True)
```

目标函数:

$$f = ||\, max(\, XW,) \,-\, Y \,||_F^2$$

手推写出已下表达式，并用 Pytorch 进行实现。

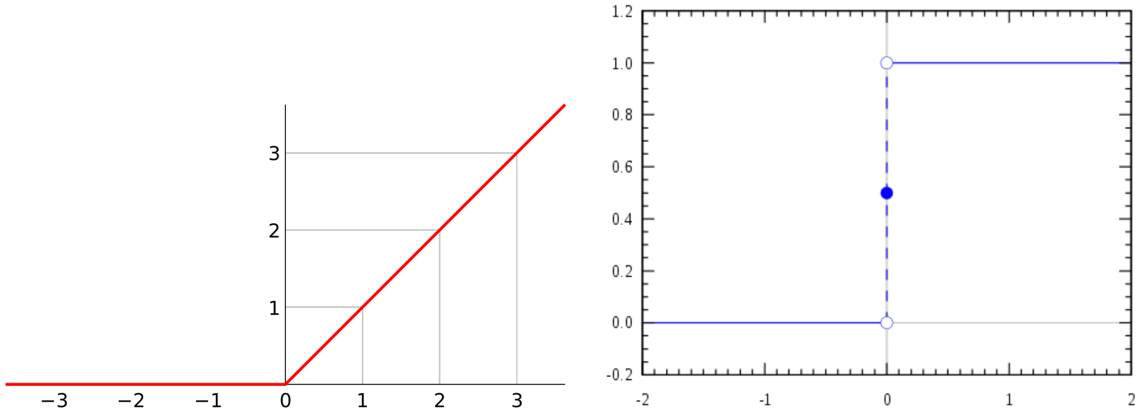
$$(1) \quad \frac{\partial f}{\partial W} \quad (2) \quad \frac{\partial f}{\partial X} \quad (3) \quad \frac{\partial f}{\partial Y}$$

2 数学式定义与程式码的数学意义说明

(1) 斜变函数与单位阶跃函数

$$f = ||\, max(\, XW,) \,-\, Y \,||_F^2$$

目标函数当中的 max(x,0) 表示的是单位斜变函数, 也就是所谓的整流线性单位函式（Rectified Linear Unit, ReLU），该函数特性在小于时归零，大于零保持原值，但当进行微分时，会变成单位阶跃函数 (Heaviside step function)，函数小于零时为零，大于零时则为一，其两者数学图形如下 (来源为 Wikipedia)。



(i) 单位斜变函数 (ii) 单位阶跃函数

Fig. 1. 单位斜变函数与单位阶跃函数

而单位斜变函数在此表示为 $R\left(x\right)$，单位阶跃函数函数在此表示为 $H\left(x\right)$，同时两者各自的函数范围与二者之间的微分关系如下:

$$R\left(x\right)=\begin{cases} x, & x \quad \geq \quad 0 \\ 0, & x \quad < \quad 0 \end{cases} \quad ; \quad H\left(x\right)=\begin{cases} 0, & x \quad < \quad 0 \\ 1, & x \quad \geq \quad 0 \end{cases} \quad ; \quad R'\left(x\right)=H\left(x\right), \quad if \quad x\neq 0$$

(2) 程式码中的数学意义

```
import torch
torch.manual_seed(0)
x = torch.randn(10,4, requires_grad=True)
W = torch.randn(4,4, requires_grad=True)
y = torch.randn(10,4, requires_grad=True)
```

程式码当中的 W 、x、y 在数学上分别代表了 W 、X、Y 三个矩阵，W $\boxed{\text{F}}$ 4 X 4 大小的矩阵，X 与 Y 矩阵皆为 10 X 4 大小的矩阵，而 Pytorch 则会随机产生矩阵中的值。

$$X = \begin{bmatrix} x_{11} & \cdots & x_{14} \\ \vdots & \ddots & \vdots \\ x_{10\ 1} & \cdots & x_{10\ 4} \end{bmatrix} \quad , \quad Y = \begin{bmatrix} y_{11} & \cdots & y_{14} \\ \vdots & \ddots & \vdots \\ y_{10\ 1} & \cdots & y_{10\ 4} \end{bmatrix} \quad , \quad W = \begin{bmatrix} w_{11} & \cdots & w_{14} \\ \vdots & \ddots & \vdots \\ w_{41} & \cdots & w_{44} \end{bmatrix}$$

3 数学推导证明

$$f = ||\, max(\, XW,) \, - \, Y\, ||_F^2 \rightarrow f \, = tr((max(XW,0) - \, Y)^T.(max(XW,0) - \, Y))$$

$$\therefore \quad df \, = tr(-dY^T.(max(XW,0) - \, Y) - (max(XW,0) - \, Y)^T.dY)$$

$$= tr(-2(max(XW,0) - \, Y)^T.dY)$$

$$\therefore \quad \frac{df}{dY} \, = \, -2(max(XW,0) - \, Y)$$

$$\because \quad f \, = tr((XW \odot \varepsilon(XW) - \, Y)^T.(XW \odot \varepsilon(XW) - \, Y))$$

其中 ε 函数，在此也就表示上节所述的单位斜变函数与单位阶跃函数，$\varepsilon` = 0$，而符号 $\odot$ 则表示逐元素对应相乘。

$$\therefore \quad df \, = tr((XdW \odot \varepsilon(XW))^T.(max(XW,0) - \, Y) + (max(XW,\, 0) - Y)^T.(XdW \odot \varepsilon(XW))$$

$$= tr(2(max(XW,\, 0) - Y)^T.\varepsilon(XW) \odot XdW)$$

$$= tr(2((max(XW,\, 0) - Y) \odot \varepsilon(XW))^T.XdW)$$

$$\therefore \quad \frac{df}{dW} \, = \, 2X^T((max(XW,0) - \, Y) \odot \varepsilon(\mathrm{XW}))$$

$$df \, = tr(((max(XW,\, 0) - Y) \odot \varepsilon(XW))^T.dX.W)$$

$$= tr(2W((max(XW,\, 0) - Y) \odot \varepsilon(XW))^T.dX)$$

$$\therefore \quad \frac{df}{dX} \, = \, 2((max(XW,0) - \, Y) \odot \varepsilon(\mathrm{XW})).W^T$$

4 Pytorch 程式码实现

程式码可以在 GitHub 项目 (kancheng/kan-cs-report-in-2021) 找到，详见 math.ipynb 档案。

(1) Pytorch 实验资料

下列为 Pytorch 所产生矩阵实验资料。

```python
import torch
torch.manual_seed(0)
x = torch.randn(10,4, requires_grad=True)
W = torch.randn(4,4, requires_grad=True)
y = torch.randn(10,4, requires_grad=True)
print(x)
print(y)
print(W)
```

```
In [2]: x
Out[2]: tensor([[-1.1258, -1.1524, -0.2506, -0.4339],
                [ 0.8487,  0.6920, -0.3160, -2.1152],
                [ 0.3223, -1.2633,  0.3500,  0.3081],
                [ 0.1198,  1.2377,  1.1168, -0.2473],
                [-1.3527, -1.6959,  0.5667,  0.7935],
                [ 0.5988, -1.5551, -0.3414,  1.8530],
                [-0.2159, -0.7425,  0.5627,  0.2596],
                [-0.1740, -0.6787,  0.9383,  0.4889],
                [ 1.2032,  0.0845, -1.2001, -0.0048],
                [-0.5181, -0.3067, -1.5810,  1.7066]], requires_grad=True)

In [3]: W
Out[3]: tensor([[ 0.2055, -0.4503, -0.5731, -0.5554],
                [ 0.5943,  1.5419,  0.5073, -0.5910],
                [-1.3253,  0.1886, -0.0691, -0.4949],
                [-1.4959, -0.1938,  0.4455,  1.3253]], requires_grad=True)

In [4]: y
Out[4]: tensor([[ 1.5091,  2.0820,  1.7067,  2.3804],
                [-1.1256, -0.3170, -1.0925, -0.0852],
                [ 0.3276, -0.7607, -1.5991,  0.0185],
                [-0.7504,  0.1854,  0.6211,  0.6382],
                [-0.0033, -0.5344,  1.1687,  0.3945],
                [ 1.9415,  0.7915, -0.0203, -0.4372],
                [-0.2188, -2.4351, -0.0729, -0.0340],
                [ 0.9625,  0.3492, -0.9215, -0.0562],
                [-0.6227, -0.4637,  1.9218, -0.4025],
                [ 0.1239,  1.1648,  0.9234,  1.3873]], requires_grad=True)
```

Fig. 2. Pytorch 矩阵

(2) 直接微分求导

下列为 Pytorch 程式码，此程式码根据目标函数 $f = ||max(XW, 0) - Y||_F^2$ 与相关数学式 $f = ||\hat{Y} - Y||_F^2$、$\hat{Y} = max(Z, 0)$、$Z = XW$ ，來进行微分求导。

```
# f = (torch.clamp(x.mm(W), 0) - y).pow(2).sum()
f1 = (torch.clamp(x.mm(W), 0) - y).pow(2).sum()
# torch.clamp 讓小於零的值，賦值🄵零。
print(f1)

# XW 矩陣乘法
z = x.mm(W)
print(z)
# 測試 torch 寫法
# test = torch.mm( x, W)
# print(test)

# ReLU
m = torch.nn.ReLU()
tm = m(z)
y_hat = tm
# 建立第二次式
f2 = (y_hat - y).pow(2).sum()

print(f2)

# W.grad.zero_()
print(W.grad)

# f.backward()
f2.backward()

print(W.grad)
print(y.grad)
print(x.grad)
```

下列为 Pytorch 程式码，根据目标函数所产生的微分求导结果。

直接微分求導

In [8]:
```
print(W.grad)
print(y.grad)
print(x.grad)
```

```
tensor([[ 18.2980,   2.7573,   2.3914,  -0.1974],
        [ 11.0817,   6.6428,   2.5163, -20.3225],
        [ -8.6662,   3.4506,  -1.8979,  -3.3608],
        [-21.1681,  -6.6739,  -1.0693,  27.0278]])
tensor([[ 2.8885e+00,  4.1639e+00,  3.4134e+00,  3.0501e+00],
        [-1.0589e+01, -2.7045e+00, -2.1849e+00, -1.7039e-01],
        [ 6.5523e-01, -1.5214e+00, -3.1982e+00, -1.5687e+00],
        [-1.5009e+00, -3.8551e+00,  4.9843e-01,  1.2764e+00],
        [-6.6077e-03, -1.0689e+00,  1.8791e+00, -4.2604e+00],
        [ 3.8829e+00,  1.5830e+00, -4.0504e-02, -7.2968e+00],
        [-4.3767e-01, -4.8701e+00, -1.4583e-01, -1.3166e+00],
        [ 1.9250e+00,  6.9834e-01, -1.8429e+00, -1.4750e+00],
        [-5.0359e+00, -9.2744e-01,  3.8436e+00, -8.0509e-01],
        [ 2.4780e-01,  2.3296e+00, -1.7491e-01, -4.2519e+00]])
tensor([[  1.1002,   0.0860,   5.3377,   0.2788],
        [  0.9583,  10.4633, -13.5234, -16.3639],
        [ -0.8712,  -0.9272,  -0.7764,   2.0790],
        [ -1.4504,   5.6914,   0.7613,  -0.9693],
        [ -1.2892,  -3.4714,  -1.9788,   4.8091],
        [ -4.0523,  -4.3127,  -3.6114,   9.6703],
        [ -0.7312,  -0.7782,  -0.6516,   1.7449],
        [ -0.8191,  -0.8718,  -0.7300,   1.9547],
        [  1.0350,   2.9930,  -6.6743,  -7.5333],
        [ -2.4616,  -2.4243,  -2.1164,   5.7128]])
```

Fig. 3. Pytorch 直接求导

(3) 公式推导求导

下列为 Pytorch 程式码，此程式码根据目标函数 $f = ||max(XW, 0) - Y||_F^2$ 与相关数学式 $f = ||\hat{Y} - Y||_F^2$、$\hat{Y} = max(Z, 0)$、$Z = XW$，跟前章数学推导后的公式进行求导。

```
y_grad = -2*(y_hat-y)
print(y_grad)

v = abs(x.mm(W)* 0)
g = torch.heaviside(input =x.mm(W), values = v)
x_grad = 2*(torch.mul((y_hat-y),g)).mm(torch.t(W))
print(x_grad)

W_grad = 2*torch.t(x).mm(torch.mul((y_hat-y),g))
print(W_grad)
```

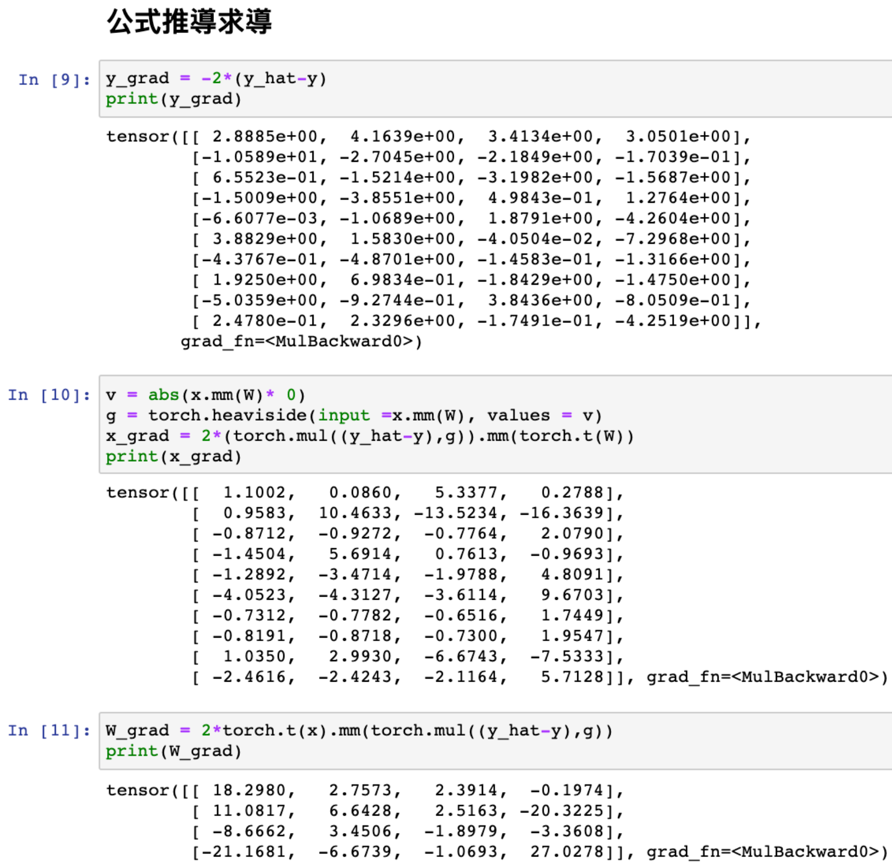


Fig. 4. Pytorch 公式求导