HW2_part3_topK_recall_precision

February 20, 2019

0.1 HW2 Solution Part 3

step1: get keypoints for each images
step2: get patches for each keypoints
step3: get descriptions for each keypoints (by forward pass patches to the network)
step4: caculate the similarity matrices
step5: get topK similar images for each query
step6: draw recall vs precision curves

```
In [1]: import torch
        import matplotlib
        import numpy as np
        import matplotlib.pyplot as plt
        %matplotlib inline
        similarity_one2one_dir = "similarity_one2one.pt"
        similarity_many2many_dir = "similarity_many2many.pt"
        similarity_one2one = torch.load(similarity_one2one_dir)
        similarity_many2many = torch.load(similarity_many2many_dir)

In [2]: print(similarity_one2one.shape, similarity_many2many.shape)
        print(similarity_one2one)
        print(similarity_many2many)

torch.Size([35, 140]) torch.Size([35, 140])
tensor([[16.1790, 16.3082, 15.7836,  ...,  9.3127,  9.5184,  9.5956],
        [ 9.5133,  9.2998,  9.1092,  ...,  9.4650,  9.3409,  9.5613],
        [ 9.9888,  9.8014,  9.6612,  ...,  9.2703,  9.4308,  9.6083],
        ...,
        [ 9.5703,  9.2934,  9.3613,  ...,  9.5277,  9.3376,  9.9001],
        [ 9.5431,  9.3677,  9.3790,  ...,  9.1698,  9.1573,  9.5070],
        [ 9.8322,  9.4572,  9.5463,  ..., 10.4304, 17.6300, 12.4188]])
tensor([[17.7728, 18.9886, 15.8135,  ...,  0.0000,  0.0000,  0.4425],
        [ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.0000],
        [ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.0000],
        ...,
        [ 0.4637,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.4748],
```

```
        [ 0.0000,  0.0000,  0.0000,  ...,  0.0000,  0.0000,  0.0000],
        [ 1.3566,  0.0000,  0.0000,  ...,  0.0000, 19.1443,  6.6333]])
```

In [3]:
```
topk_one2one_ind   = torch.zeros(4, 35, 4)  # k, 35, k
topk_many2many_ind = torch.zeros(4, 35, 4)  # k, 35, k

for k in range(4):
    _, topk_one2one   = torch.topk(similarity_one2one, k+1, dim = 1)
    _, topk_many2many = torch.topk(similarity_many2many, k+1, dim = 1)

    topk_one2one_ind[k, :, :k+1] = topk_one2one + 1
    topk_many2many_ind[k, :, :k+1] = topk_many2many + 1
```

In [4]:
```
# read gt
gt_dir = "image_retrieval/ground_truth.txt"
gt = torch.empty(35, 4)
with open(gt_dir, 'r') as f:
    point = 0
    for line in f:
        if not line.startswith("q"):
            continue
        one_line = line.strip().split("  ")
        query_idx = int(one_line[0][1:])
        image_idx = int(one_line[1])
        gt[query_idx-1][point%4] = image_idx
        point+=1
```

In [5]:
```
P_one2one = torch.zeros(4,1)
R_one2one = torch.zeros(4,1)
for k in range(4):
    P_1, R_1 = 0, 0
    for idx in range(35):
        for topk_idx in topk_one2one_ind[k][idx][:k+1]:
            if int(topk_idx) in gt[idx]:
                P_1 += 1/(k+1)/35 # sum_P /Q
                R_1 += 1/4/35
    P_one2one[k] = P_1
    R_one2one[k] = R_1

P_many2many = torch.zeros(4,1)
R_many2many = torch.zeros(4,1)
for k in range(4):
    P_1, R_1 = 0, 0
    for idx in range(35):
        for topk_idx in topk_many2many_ind[k][idx][:k+1]:
            if int(topk_idx) in gt[idx]:
                P_1 += 1/(k+1)/35 # sum_P /Q
```

```
                    R_1 += 1/4/35
            P_many2many[k] = P_1
            R_many2many[k] = R_1
```

```
In [6]: print(P_one2one[:,0])
        print(R_one2one[:,0])
        print(P_many2many[:,0])
        print(R_many2many[:,0])
```

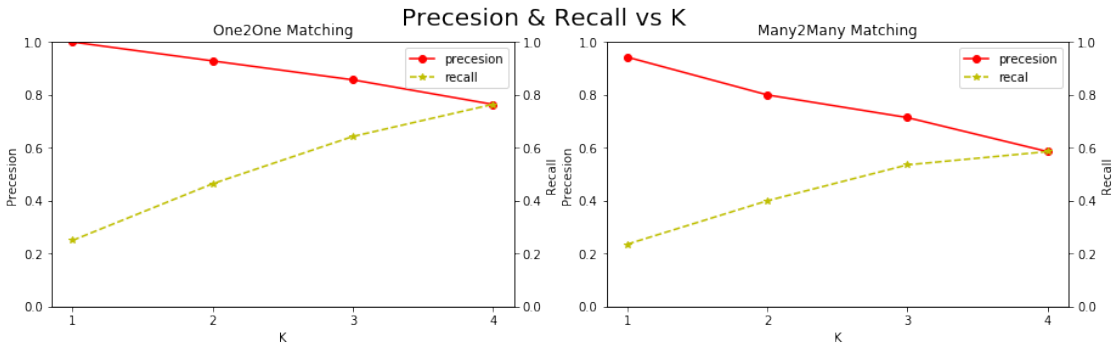
```
tensor([1.0000, 0.9286, 0.8571, 0.7643])
tensor([0.2500, 0.4643, 0.6429, 0.7643])
tensor([0.9429, 0.8000, 0.7143, 0.5857])
tensor([0.2357, 0.4000, 0.5357, 0.5857])
```

0.1.1 Visualize Result

```
In [7]: k = [i for i in range(1, 5)]
        fig = plt.figure(figsize=(15, 4))
        plt.suptitle('Precesion & Recall vs K', fontsize=20)
        # fig.subplots_adjust(mid=0.2)
        # fig.tight_layout()
        ax1 = fig.add_subplot(121)
        plt.title('One2One Matching')
        plt.xticks([1,2,3,4])
        ax1.set_ylim(0,1)
        lns1 = ax1.plot(k, P_one2one[:,0].numpy(), "-ro",label="precesion")
        ax2 = ax1.twinx()
        ax2.set_ylim(0,1)
        lns2 = ax2.plot(k, R_one2one[:,0].numpy(),"--y*",label="recall")

        ax1.set_ylabel('Precesion')
        ax2.set_ylabel('Recall')
        ax1.set_xlabel('K')
        lns = lns1+lns2
        labs = [l.get_label() for l in lns]
        ax1.legend(lns, labs, loc=0)



        ax3 = fig.add_subplot(122)
        plt.title('Many2Many Matching')
        plt.xticks([1,2,3,4])
        ax3.set_ylim(0,1)
        lns3 = ax3.plot(k, P_many2many[:,0].numpy(), "-ro",label="precesion")
        ax4 = ax3.twinx()
        ax4.set_ylim(0,1)
        ax3.set_ylabel('Precesion')
```

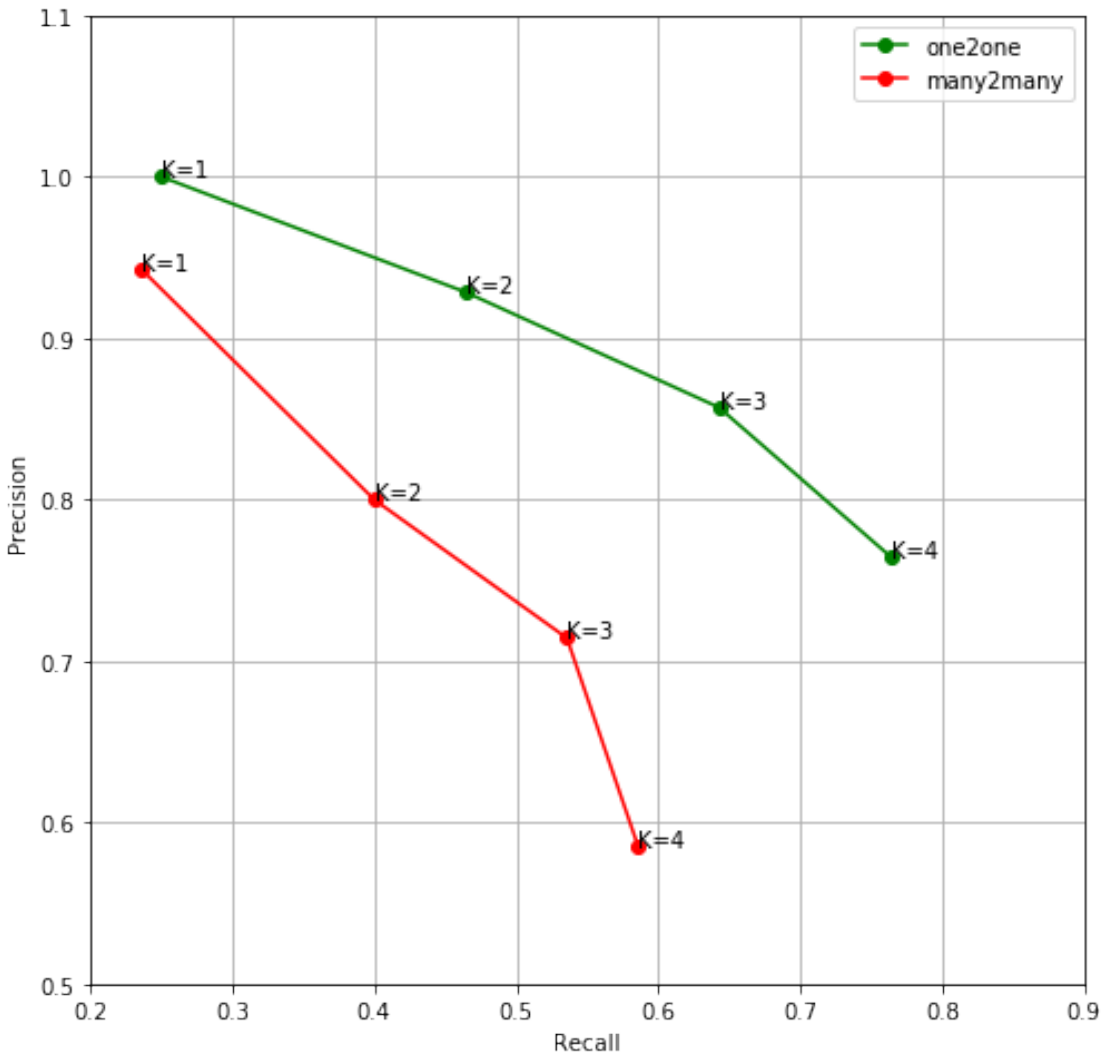
```
        ax4.set_ylabel('Recall')
        ax3.set_xlabel('K')

        # ax2.axis([0, 1, 0, 1])
        lns4 = ax4.plot(k, R_many2many[:,0].numpy(),"--y*",label="recal")

        lnss = lns3+lns4
        labss = [h.get_label() for h in lnss]
        ax3.legend(lnss, labss, loc=0)
```

Out[7]: <matplotlib.legend.Legend at 0x7f9946767b38>



In [8]:
```
        k = [i for i in range(1, 5)]
        fig = plt.figure(figsize=(8, 8))
        plt.suptitle('Precesion vs Recall', fontsize=20)
        ax1 = fig.add_subplot(111)
        ax1.set_xlim(0.2,0.9)
        ax1.set_ylim(0.5,1.1)
        ax1.set_ylabel('Precision')
        ax1.set_xlabel('Recall')
        ax1.grid(True)
        lns1 = ax1.plot(R_one2one[:,0].numpy(), P_one2one[:,0].numpy(), "-go",label="one2one")
        for i in range(4):
            ax1.annotate('K={}'.format(i+1), xy=(R_one2one[i,0].numpy(), P_one2one[i,0].numpy())
            ax1.annotate('K={}'.format(i+1), xy=(R_many2many[i,0].numpy(), P_many2many[i,0].nump
        lns2 = ax1.plot(R_many2many[:,0].numpy(), P_many2many[:,0].numpy(), "-ro",label="many2ma
        lnss = lns1+lns2
        labss = [h.get_label() for h in lnss]
        ax1.legend(lnss, labss, loc=0)
```

Out[8]: <matplotlib.legend.Legend at 0x7f99466c0eb8>


Precesion vs Recall
one2one
many2many
K=1
K=2
K=3
K=4
K=1
K=2
K=3
K=4
Precision
Recall
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9