Exercise mean-shift-cow implementation

Qi Ma, 20-960-225, qimaqi@student.ethz.ch

November 1, 2021

1 Mean-shift algorithm

Since we consider the radius to be $+\infty$ so in distance function we calculate given point or batch with all points 2D distance

```python
def distance(x, X):
    # input x, 1x3 tensor
    # input X, 3675x3 tensor
    # output dist, 3675 tensor
    dist = torch.norm(x-X,dim=1)
    return dist
```

Figure 1: distance based on point update

```python
def distance_batch(x, X):
    # input x: batch_size x 3
    # input x: 3675x3 data
    # output dist: batch_size x 3675
    dist = torch.norm(X-x[:,None],dim=2)
    return dist
```

Figure 2: distance based on batch update

In gaussian function I use gaussian kernel with 2.5 bandwidth

```python
def gaussian(dist, bandwidth):
    # input dist 3675
    # input bandwith constant
    # output weight: 3675
    bandwidth = torch.tensor(bandwidth).double()
    weight = (-torch.square(dist)/(2*torch.square(bandwidth))).exp()
    return weight
```

Figure 3: Gaussian kernel weight

Then by adopting point and batch update we achieve speed from **69.85 s** to **47.28 s**. With 32% gain by adopting batch size 32 and num of work 4.

```python
def update_point(weight, X):
    # input weight 3675
    # input X 3675x3
    # output
    xi = torch.matmul(X.t(), weight)/torch.sum(weight)
    return xi
```

Figure 4: update based on point update

```python
def update_point_batch(weight, X):
    # weight batch_size x 3675
    # X: 3675x3
    # output bacth_size x 3
    xi = torch.matmul(weight, X)
    norm_weight = torch.sum(weight,dim=1)
    y_ = torch.matmul(torch.inverse(torch.diag(norm_weight)), xi)
    return y_
```

Figure 5: update based on batch update

2 Segnet

The validation result of Segnet is 0.8729

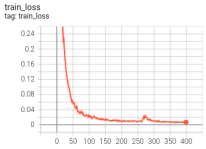

Figure 6: Segnet training loss

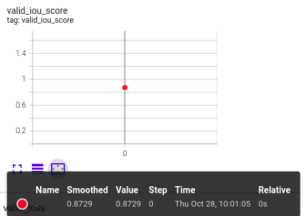

Figure 7: Segnet validation loss