20171682 임정호

1. 사용한 함수

우선 예시로 주어졌던 이미지화하는 코드를 살펴보았더니, 매개변수인 images가 64차원의 데이터가 들어가야 reshape(8,8)에서 에러가 나지 않는 것을 봤고, 복원된 데이터를 넣으면 되는 것을 알았다. 그래서 32,4,3,2차원에서 쓰이므로 함수로 만들었다.

In [29]:
```
def show_image(images):
    _, axes = plt.subplots(nrows=1, ncols=4, figsize=(10, 3))
    for ax, image in zip(axes, images):
        ax.set_axis_off()
        image = image.reshape(8, 8)
        ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
```

todo 2에서 구현한 pca함수를 통해 n_components의 차원으로 축소시킨 데이터를 다시 복원해야 한다. 원래 데이터는 64차원, 64개의 column을 가지고 있는 상태였고, 차원 축소된 데이터는 n_components만큼의 colum을 가지고 있는 상태이다. 1797개의 데이터가 있는 것을 확인했고, 1797 X n을 다시 1797 x 64로 변환해야 한다는 것이다. 그러기 위해서는 n차원으로 축소시킬 때 사용했던 eigen vector를 다시 사용해야 한다. 그리고 축소시킬 때 각 column의 평균을 빼주는 작업을 수행했었는데, 원래의 데이터처럼 복구하려면 빼준 평균을 다시 더해줘야 한다. 그래서 사용했던 eigen vector와 빼준 평균들을 리턴해주는 get_compo_aver()라는 함수를 구현하여 사용하였다.

In [30]:
```
def get_compo_aver(X,n_components):

    norm_X = X-np.mean(X,axis=0)

    cov_norm_X = np.cov(norm_X.T)

    eigen_val, eigen_vec = np.linalg.eig(cov_norm_X)

    eigen_vec = eigen_vec.T

    idxs = np.argsort(eigen_val)[::-1]

    new_eigen_vec = eigen_vec[idxs]

    return new_eigen_vec[:n_components],X-norm_X
```

그리고 각 차원마다 원래의 데이터와 복원한 데이터의 MSE값을 구하여 출력하게끔 출력 예시가 있어서, 출력하는 함수를 구현했다.

n [31]:
```
def get_mse(dim,original,recovered):

    mse = sum(np.square(original-recovered).mean(axis=0))
    print(f"{dim}차원: MSE error: {mse}")
```

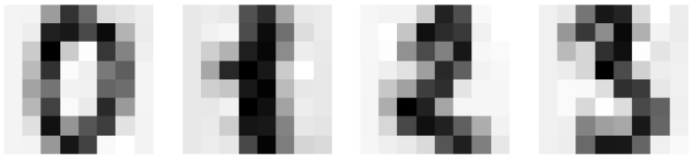
2. 복원 결과

1) 32차원

In [32]:
```
reduced = student_pca(images, 32)
compo,aver = get_compo_aver(images,32)
recovered = np.dot(reduced,compo)+aver

get_mse(32,images,recovered)
show_image(recovered)
```

```
32차원: MSE error: 40.42470493509362
```



2) 4차원

In [33]:
```
reduced = student_pca(images, 4)
compo,aver = get_compo_aver(images,4)
recovered = np.dot(reduced,compo)+aver

get_mse(4,images,recovered)
show_image(recovered)
```

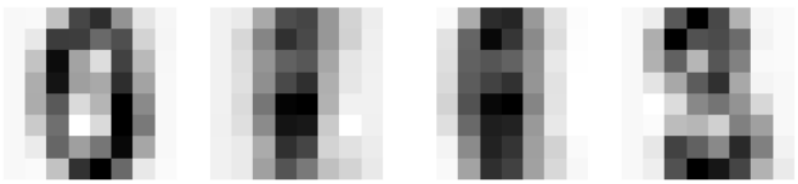
```
4차원: MSE error: 616.1911300562693
```

3) 3차원

In [34]:
```
reduced = student_pca(images, 3)
compo,aver = get_compo_aver(images,3)
recovered = np.dot(reduced,compo)+aver

get_mse(3,images,recovered)
show_image(recovered)
```

```
3차원: MSE error: 717.2352446162666
```



4) 2차원

```
In [35]: reduced = student_pca(images, 2)
         compo,aver = get_compo_aver(images,2)
         recovered = np.dot(reduced,compo)+aver

         get_mse(2,images,recovered)
         show_image(recovered)
```

```
2차원: MSE error: 858.9447808487331
```

3. 결과 분석

차원을 축소시킨다는 자체가 사용하는 데이터의 정보를 줄인다는 의미이다. 따라서 축소를 한다는 것은 결국 정보의 유실이 필연적으로 발생한다. 다만, 불필요한 정보를 줄일 수 있지만, 지나치게 차원을 축소하다보면 각 label을 구분 짓던 feature들의 사라져 결국 복원했을 때 제대로 식별하기 어려운 문제가 발생한다. 위 결과또한 마찬가지이다. 32차원으로 축소했다가 복원한 데이터를 이미지화했을 때는 숫자를 식별가능할 수 있었다. 하지만 4차원부터 1과 2를 식별하기 어려워졌음을 알 수 있다. 더 많은 차원을 축소하는 작업은 더 많은 데이터의 손실을 감수하면서 차원을 축소하는 것이므로, 32부터 2차원으로 갈 수록 원래 데이터와의 오차율이 증가하는 것을 확인할 수 있다.