고소실3 과제 보고서

20171693 조병화

이번 과제에서는 기존의 Digit 데이터를 To do 2 에서 만든 student_pca함수를 이용해 차원을 축소시키고, 이를 reconstruction을 통해 복원시킨다. 이 후 복원 된 이미지를 시각화 했을 때 오차율을 비교해 보았다.

우선, 축소할 차원을 밑의 식에 기입해 차원을 축소시킨다.

pca_student_score = student_pca(X, n_components=32)

이후, to do 2에서 차원을 축소시킨 과정을 다시 역으로 돌아가면서 차원을 복구시킨다.

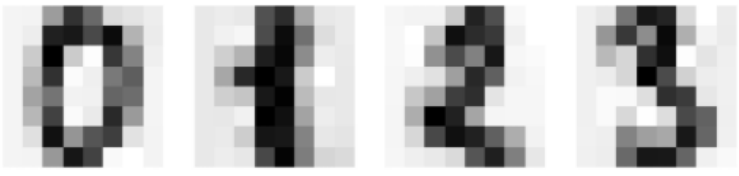
To do 2 에서 공분산 행렬을 생성하고, 그에서 eigenvalue, eigenvector을 구해 기존 데이터에 투영시켜 축소를 시켰으므로, 투영된 데이터에 transpose한 데이터를 다시 투영시켜 원래의 차원으로 돌린후, 같은 방식으로 기존의 64차원의 데이터를 얻는다. 물론 이 때 eigenvalue를 32개만 사용하였기 때문에 기존의 64차원과 같아질 수 는 없고, 마찬가지의 이유로 eigenvalue를 가장 가중치가 큰 것부터 사용하는 개수가 줄어들수록 오차율은 커질 수 밖에 없다.

```
tran_pca_results = pca_student_score.T
new_X_meaned = (eigenvector_subset.dot(tran_pca_results)).T
new_X = new_X_meaned + np.mean(X, axis = 0)
```

위 과정을 통해 다시 복구된 이미지를 차원에 대해 mse오차율을 비교해 보았다.

```
In [1661]: _, 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')
loss = np.sum((X - new_X)**2, axis=1).mean()
print(loss)

40.42470493509364
```



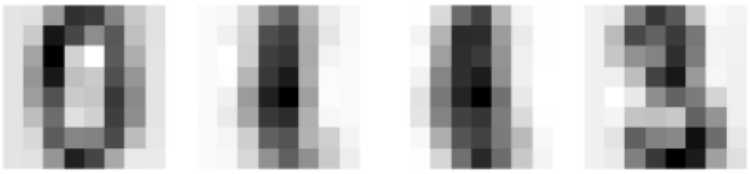
기존 64차원을 32차원으로 축소시킨 후 다시 복구시켰을 때 원래의 이미지와의 차이를 비교한 값이다. 40정도가 나온다.

다음은 4차원, 3차원, 2차원으로 축소시킨 결과를 보도록 하자.

In [1663]:
```
_, 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')

loss = np.sum((X - new_X)**2, axis=1).mean()
print(loss)
```

```
616.1911300562696
```

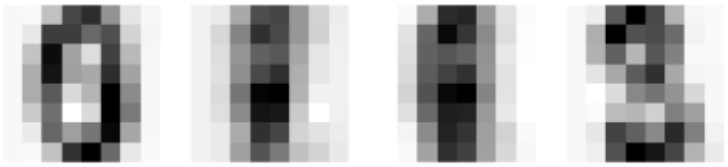


4차원으로 축소후 복원했을 때

In [1666]:
```
_, 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')

loss = np.sum((X - new_X)**2, axis=1).mean()
print(loss)
```

```
717.2352446162665
```

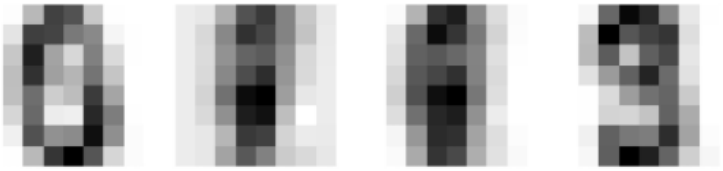


3차원으로 축소후 복원했을 때

In [1668]:
```
_, 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')

loss = np.sum((X - new_X)**2, axis=1).mean()
print(loss)
```

```
858.9447808487328
```



2차원으로 축소후 복원했을 때

차원을 많이 줄였다 복원할수록 위의 예상과 같이 오차율이 매우 커짐을 확인할 수 있다.

32차원에서 4차원으로의 차이가 워낙 커 16차원으로 축소후 복원했을 때의 결고도 확인 해 보았다.

In [1731]:
```
_, 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')

loss = np.sum((X - new_X)**2, axis=1).mean()
print(loss)
```

```
180.93970325737862
```