

Rules:

- Unless otherwise stated, all answers must be mathematically justified.
- Partial answers will be graded.
- Your submission has to be uploaded to Gradescope. In Gradescope, indicate the page on which each problem is written.
- You can work in groups but each student must write his/her/their own solution based on his/her/their own understanding of the problem. Please list on your submission the students you work with for the homework (this will not affect your grade).
- Problems with a (\star) are extra credit, they will not (directly) contribute to your score of this homework. However, for every 4 extra credit questions successfully answered your lowest homework score get replaced by a perfect score.
- If you have any questions, feel free to ask them on Ed Discussion (so that everyone can benefit from the answer) or stop at the office hours.

Problem 7.1 (3 points). We say that a symmetric matrix $M \in \mathbb{R}^{n \times n}$ is positive semi-definite if for all $x \in \mathbb{R}^n$, $x^\top M x \geq 0$.

- (a) Let $D \in \mathbb{R}^{n \times n}$ be a diagonal matrix. When is D positive semi-definite?

We first give the claim: When all the diagonal entries are bigger than or equal to zero, D is PSD.

Proof. $\forall \vec{x} \in \mathbb{R}^n$, we have $\vec{x}^\top D \vec{x} = \sum_{i=1}^n D_{i,i} x_i^2 \geq 0$, which implies that D is PSD. \square

- (b) Let $M \in \mathbb{R}^{n \times n}$ be a symmetric matrix. Show that M is positive semi-definite **if and only if** its eigenvalues are all non-negative.

Proof.

(a) (\implies): Since M is symmetric, we know from the spectrum theorem that $M = P D P^\top$ for some orthogonal matrix P and diagonal matrix D . Then we have that $\vec{x}^\top P D P^\top \vec{x} = \vec{y}^\top D \vec{y}$ where $\vec{y} = P^\top \vec{x}$. Since $\vec{y}^\top D \vec{y} = \sum_{i=1}^n D_{i,i} y_i^2 = \sum_{i=1}^n \lambda_i y_i^2 \geq 0$ by definition of PSD. Since the above inequality holds for all $\vec{y} \in \mathbb{R}^n$, we have that $\lambda_i \geq 0$.

(b) (\impliedby): If all eigenvalues of M are non-negative, we have $\lambda_i \geq 0, \forall i = 1, 2, \dots, n$ (Note that there could be some i, j such that $\lambda_i = \lambda_j$). We have that $\forall \vec{x} \in \mathbb{R}^n$, $\vec{x}^\top M \vec{x} = \vec{y}^\top D \vec{y} = \sum_{i=1}^n \lambda_i y_i^2$ as shown in the forward direction. Since $\lambda_i \geq 0$, we have $\sum_{i=1}^n \lambda_i y_i^2 \geq 0$, which implies that $\vec{x}^\top M \vec{x} \geq 0, \forall \vec{x} \in \mathbb{R}^n$, which implies that M is PSD. \square

- (c) Let $A \in \mathbb{R}^{n \times m}$ be any rectangular matrix. Show that $A^\top A$ and $A A^\top$ are positive semi-definite. (This shows that these matrices have non-negative eigenvalues.)

(a) For $A^\top A$ we have $\forall \vec{x} \in \mathbb{R}^n$, $\vec{x}^\top A^\top A \vec{x} = \|A \vec{x}\|_2^2 \geq 0$, which implies that $A^\top A \in \mathbb{S}_+^n$.

(b) For $A A^\top$ we have $\forall \vec{x} \in \mathbb{R}^n$, $\vec{x}^\top A A^\top \vec{x} = \|A^\top \vec{x}\|_2^2 \geq 0$, which implies that $A A^\top \in \mathbb{S}_+^n$.

Problem 7.2 (3 points). Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix.

- (a) Let $A = PDP^\top$ with P orthogonal and D diagonal. Show that for any $k \in \mathbb{N}$, $A^k = PD^kP^\top$.

We show it through mathematical induction, where the inductive hypothesis is $P(n) : A^n = PD^nD^\top$.

Proof.

(a) Base Step: When $k = 0$, we have $I = A^0 = PIP^\top = I$, which is true.

(b) Inductive Step: Suppose $P(k), k \geq 0$ is true, then $A^k = PD^kP^\top$. Then for $n = k + 1$, $A^{k+1} = PD^kP^\top PDP^\top = PD^{k+1}P^\top$, which shows that $P(k + 1)$ is true.

By mathematical induction, we have that $P(n)$ is true $\forall n \in \mathbb{N}$. □

- (b) Same question when k is a negative integer.

We show it through mathematical induction, where the inductive hypothesis is $P(n) : A^{-n} = PD^{-n}D^\top$.

Proof.

(a) Base Step: When $k = 1$, we have $A^{-1} = (PDP^\top)^{-1} = PD^{-1}P^\top$, which is true.

(b) Inductive Step: Suppose $P(k), k \geq 0$ is true, then $A^{-k} = PD^{-k}P^\top$. Then for $n = k + 1$, $A^{-(k+1)} = PD^{-k}P^\top PD^{-1}P^\top = PD^{-(k+1)}P^\top$, which shows that $P(k + 1)$ is true.

By mathematical induction, we have that $P(n)$ is true $\forall n \in \mathbb{N}$. □

- (c) Assume that A is positive semi-definite. Prove that there exists a symmetric positive semi-definite matrix $B \in \mathbb{R}^{n \times n}$ such that $A = B^2$. (Hint: in some sense, $B = A^{1/2}$. Can you guess how to define B ?)

The claim would be: there exists a PSD matrix $B = PD^{\frac{1}{2}}P^\top$ such that $A = B^2$ where P is the eigenvectors of A and the diagonal entries of D are eigenvalues of A .

Proof. Since A is PSD, by spectrum theorem we have $A = PDP^\top = PD^{\frac{1}{2}}P^\top PD^{\frac{1}{2}}P^\top$ and we let $B = PD^{\frac{1}{2}}P^\top$. Note that since A is PSD and we have proved that all the eigenvalues of A are non-negative, so $D^{\frac{1}{2}}$ is still a real matrix (no complex numbers on the diagonal). Moreover, $B^\top = (PD^{\frac{1}{2}}P^\top)^\top = P(D^{\frac{1}{2}})^\top P^\top = PD^{\frac{1}{2}}P^\top = B$, so B is symmetric. Since $\forall \vec{x} \in \mathbb{R}^n, \vec{x}^\top B \vec{x} = \vec{x}^\top PD^{\frac{1}{2}}P^\top \vec{x} = \vec{y}^\top D^{\frac{1}{2}} \vec{y} = \sum_{i=1}^n \lambda_i^{\frac{1}{2}} y_i^2 \geq 0$, which implies that B is PSD, which finishes our proof. □

Problem 7.3 (2 points). Consider a dataset $x_1, \dots, x_n \in \mathbb{R}^d$ with mean $\mu \in \mathbb{R}^d$ and covariance $\Sigma \in \mathbb{R}^{d \times d}$

- (a) Let $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$. Define $y_i = Ax_i + b$ for $i = 1, \dots, n$. Calculate the mean μ' and covariance Σ' of the dataset y_1, \dots, y_n .

We have the following:

$$(a) \mu' = \frac{1}{n} \sum_{i=1}^n y_i = \frac{1}{n} \sum_{i=1}^n (Ax_i + b) = \frac{1}{n} (A \sum_{i=1}^n x_i + nb) = \frac{1}{n} A \sum_{i=1}^n x_i + \frac{nb}{n} = A\mu + b.$$

(b) We have the following derivations:

$$\begin{aligned} \Sigma' &= \frac{1}{n} \sum_{i=1}^n (y_i - \mu')(y_i - \mu')^\top && \text{(By Definition)} \\ &= \frac{1}{n} \sum_{i=1}^n A(x_i - \mu)(A(x_i - \mu))^\top && \text{(From part(a))} \\ &= \frac{1}{n} \sum_{i=1}^n A(x_i - \mu)(x_i - \mu)^\top A^\top && \text{(Properties of Transpose)} \\ &= A \frac{1}{n} \sum_{i=1}^n (x_i - \mu)(x_i - \mu)^\top A^\top && \text{(Linearity of linear map)} \\ &= A \Sigma A^\top \end{aligned}$$

- (b) Find A and b as a function of μ and Σ so that $\mu' = 0$ and $\Sigma' = \text{Id}$ (there are several solutions for A , you do not have to find all of them). We call such an operation “whitening”. (Hint: search for A of the form $PD'P^\top$, where $\Sigma = PDP^\top$ using the spectral theorem.)

In other words, we have to find A, b that satisfies the following:
$$\begin{cases} A\mu + b = 0 \\ A\Sigma A^\top = I \end{cases}$$

If we let $A = PD'P^\top$ where $\Sigma = PDP^\top$ then we would have
$$\begin{cases} PD'P^\top u + b = 0 \\ PD'P^\top PDP^\top PD'P^\top = I \end{cases},$$

that is
$$\begin{cases} D'P^\top u + P^\top b = 0 & \text{, If we times } P^\top \text{ on both sides} \\ (D')^2 D = P^\top P = I & \text{, If we times } P^\top \text{ on the left side and } P \text{ on the right side} \end{cases}$$

Then one possible solution would be:
$$\begin{cases} D' = D^{-\frac{1}{2}} \\ b = -(P^\top)^{\frac{1}{2}} u \end{cases}. \text{ Note that this can be constructed since } P \text{ is invertible (orthogonal) and the solution } b \text{ came from the least square procedure where } P^\top b = -D^{-\frac{1}{2}} P^\top u.$$

Finally we assemble the result and get $A = PD^{-\frac{1}{2}}P^\top = P^{\frac{1}{2}}\Sigma^{-\frac{1}{2}}(P^\top)^{\frac{1}{2}}$ and $b = -D^{-\frac{1}{2}}u$.

Alternatively, we could construct
$$\begin{cases} A = (P\Sigma^{\frac{1}{2}})^{-1} \\ b = -(P\Sigma^{\frac{1}{2}})^{-1}u \end{cases} \text{ that satisfies the above equations.}$$

Problem 7.4 (3 points). *Complete the `mnist_pca.ipynb` Jupyter notebook to compute the mean, covariance, and PCA of the MNIST dataset. Please only submit a pdf version of your notebook (right-click → ‘print’ → ‘Save as pdf’).*

[Jupyter notebook pdf link](#)

[Jupyter notebook link](#)

In [14]:

```
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline
```

The MNIST dataset

The MNIST dataset is composed of 70,000 28×28 grayscale images of handwritten digits. It is represented as a $70000 \times 28 \times 28$ numpy array (a "3d matrix").

In [15]:

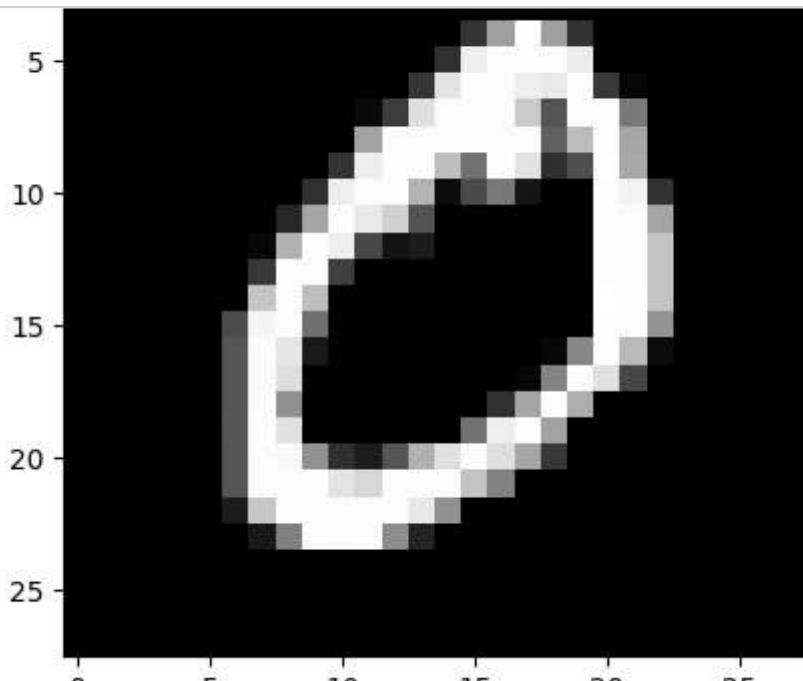
```
x = np.load("mnist.npy")
print(x.shape)
```

(70000, 28, 28)

Display the first few digits in the dataset.

In [16]:

```
for i in range(5):
    plt.imshow(x[i], cmap="gray")
    plt.show()
```



Computing and diagonalizing the covariance of MNIST

We will interpret each image as a vector in \mathbb{R}^d with $d = 28^2 = 768$. The dataset can thus be seen as a matrix $\text{in } \mathbb{R}^{n \times d}$ where $n = 70000$.

In [17]:

```
xx = x.reshape((x.shape[0], -1))
xx
```

Out[17]:

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], dtype=uint8)
```

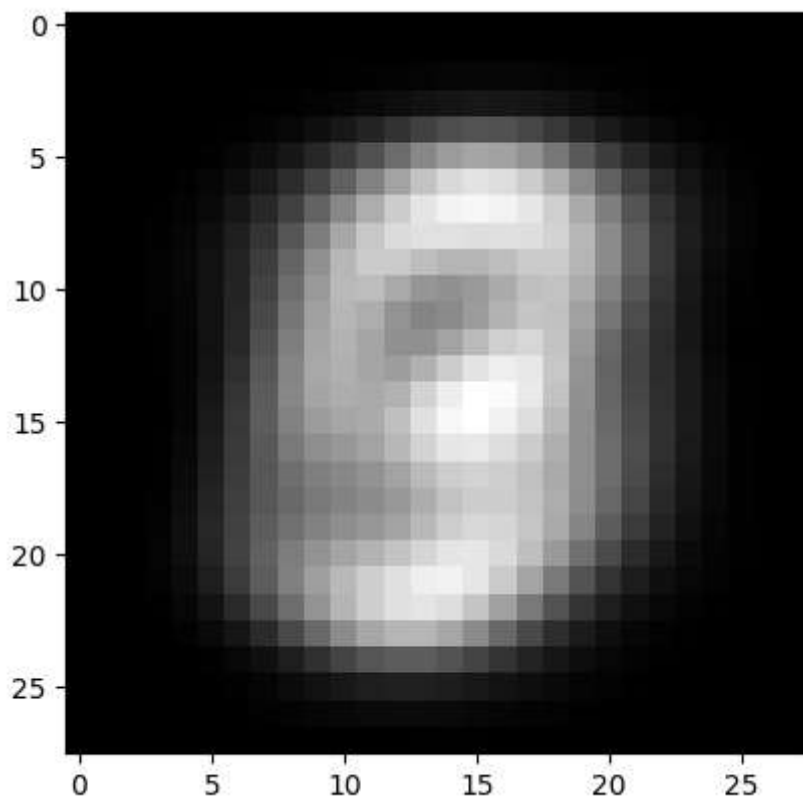
1. Compute the mean $\mu \in \mathbb{R}^d$ of the MNIST dataset and plot it as a 28×28 image.

In [18]:

```
# Your answer here
mean_vector = xx.mean(axis=0)
mean_image = mean_vector.reshape((28, 28))
plt.imshow(mean_image, cmap="gray")
```

Out[18]:

<matplotlib.image.AxesImage at 0x23ec5285fc0>



2. Compute the covariance $\Sigma \in \mathbb{R}^{d \times d}$ of the MNIST dataset and diagonalize it using the function `np.linalg.eigh`.

In [19]:

Your answer here

```
cov_matrix = (1 / xx.shape[0]) * (xx - mean_vector).T @ (xx - mean_vector)
eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)
eigenvalues, eigenvectors
```

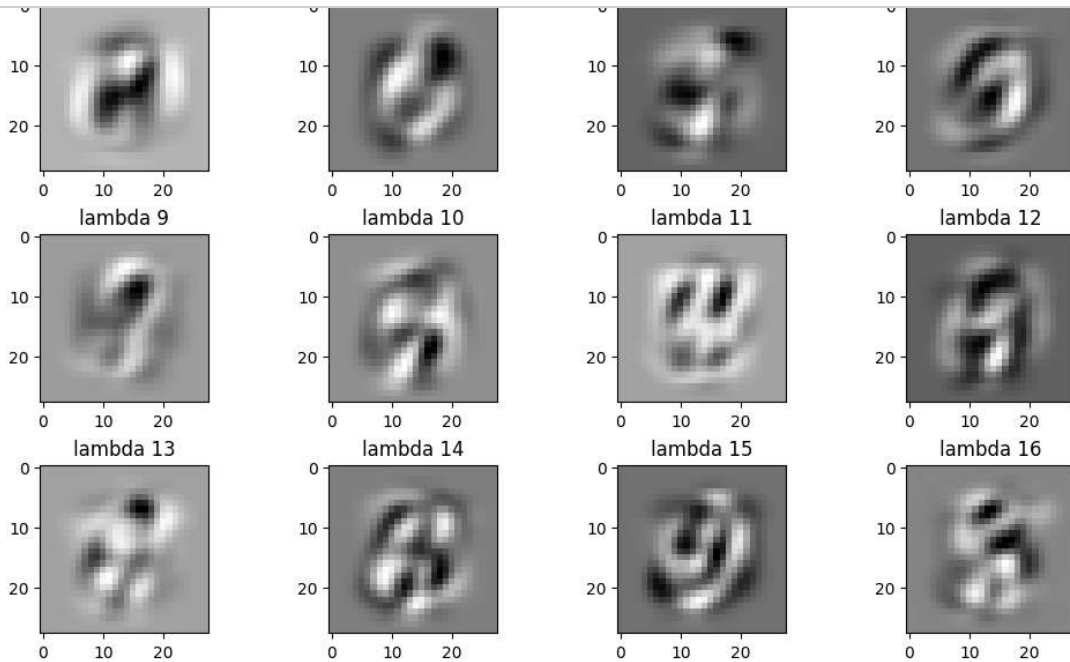
```
9.24113022e+03, 9.69015100e+03, 9.84750631e+03, 1.01692003e+04,
1.06378912e+04, 1.08680122e+04, 1.09654969e+04, 1.16184074e+04,
1.19714546e+04, 1.23968292e+04, 1.28893227e+04, 1.31610777e+04,
1.35883208e+04, 1.43447224e+04, 1.52605741e+04, 1.55847279e+04,
1.60383427e+04, 1.64280388e+04, 1.67066997e+04, 1.73116665e+04,
1.86409047e+04, 1.94395023e+04, 2.00863333e+04, 2.06079485e+04,
2.21394661e+04, 2.25055712e+04, 2.36673094e+04, 2.53908053e+04,
2.69499231e+04, 2.77770236e+04, 2.87712502e+04, 3.02965122e+04,
3.12002508e+04, 3.28986421e+04, 3.46356878e+04, 3.65648922e+04,
3.95454343e+04, 4.07231430e+04, 4.38698621e+04, 4.52536592e+04,
5.09812503e+04, 5.43096063e+04, 5.81044457e+04, 5.85518694e+04,
6.98875556e+04, 7.22588790e+04, 8.03348093e+04, 9.46111872e+04,
9.91139674e+04, 1.12443533e+05, 1.47668187e+05, 1.67689177e+05,
1.85334709e+05, 2.10927341e+05, 2.45429921e+05, 3.34289286e+05]),
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

3. Plot the ordered eigenvalues $\lambda_1 \geq \dots \geq \lambda_k \geq \dots$ as a function $k = 1, \dots, d$ with the x axis in log scale, and the first few eigenvectors u_1, \dots, u_k, \dots as 28×28 images.

In [21]:

```
# Your answer here
sorted_eigenvalues = sorted(eigenvalues, reverse=True)
sorted_eigenvectors = np.fliplr(eigenvectors)

plt.plot(range(1, len(sorted_eigenvalues)+1), sorted_eigenvalues)
fig, axes = plt.subplots(4, 4, layout='constrained', figsize=(10, 8))
for i in range(4):
    for j in range(4):
        index = 4 * i + j
        axes[i][j].imshow(sorted_eigenvectors[:, index].reshape(28, 28), cmap="gray")
        axes[i][j].set_title(f"lambda {index + 1}")
```



PCA compression of MNIST

4. Let $k \in \mathbb{N}$. Compute the k -dimensional PCA approximation z_1, \dots, z_n of the MNIST dataset using the eigenvectors u_1, \dots, u_k . Then, compute the reconstructed images $\hat{x}_i = \mu + z_{i,1}u_1 + \dots + z_{i,k}u_k$, which are equal to the mean μ plus the orthogonal projection of $x_i - \mu$ on $\text{Span}(u_1, \dots, u_k)$. Display the first 5 reconstructed images $\hat{x}_1, \dots, \hat{x}_5$. Choose a small value of k that still allows recognizing the digits.

In [10]:

```

# Your answer here
# Compute the dimension of each data point
def dim_data(data):
    return data.shape[1]

# Compute the mean of the sample
def mean_data(data):
    return np.mean(data, axis=0)

# Compute the standard deviation of the sample
def sd_data(data):
    return (data - np.mean(data, axis=0)) / np.std(data, axis=0)

# Centerize the data for further computation of the covariance matrix
def centerize_data(data):
    # Centerize the data
    return data - np.mean(data, axis=0)

# Compute the eigenbasis with k eigenvectors
def compute_eigenbasis_k(centered_data, k):
    # Compute the top k eigenvectors for our eigenbasis
    cov_matrix = 1 / len(centered_data) * centered_data.T @ centered_data

    eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)

    # Flip the columns, in reversed order.
    sorted_eigenvectors = np.fliplr(eigenvectors)[:,:k]

    return sorted_eigenvectors

# Main Procedure: PCA
def PCA_procedure(data, k):
    centered_data = centerize_data(data)
    V_k = compute_eigenbasis_k(centered_data, k)

    # Find PCA projection
    Z_k = V_k.T @ centered_data.T

    # Z_k is a matrix with (k, 70000), where the projection for each data point onto eigenbasis
    return Z_k, V_k

# Main Procedure: Inverse PCA
def inverse_PCA_procedure(Z_k, V_k, data, k):
    centered_data = centerize_data(data)

    mu = data.mean(axis=0)

    # Revert to origin
    RC_k = V_k @ Z_k + mu.reshape(dim_data(data), 1)

    # RC_k is a matrix with (784, 70000), where the reconstructed coordinates for each data point
    return RC_k

```

```

# Display the reconstructed images
def show_first_five_reconstructed(data, k):
    # Draw the first five reconstructed images
    fig, axes = plt.subplots(1, 5, figsize=(16,16), constrained_layout=True)

    Z_k, V_k = PCA_procedure(data, k)
    RC_k = inverse_PCA_procedure(Z_k, V_k, data, k)

    for i in range(5):
        axes[i].imshow(RC_k[:,i].reshape(28,28), cmap="gray")

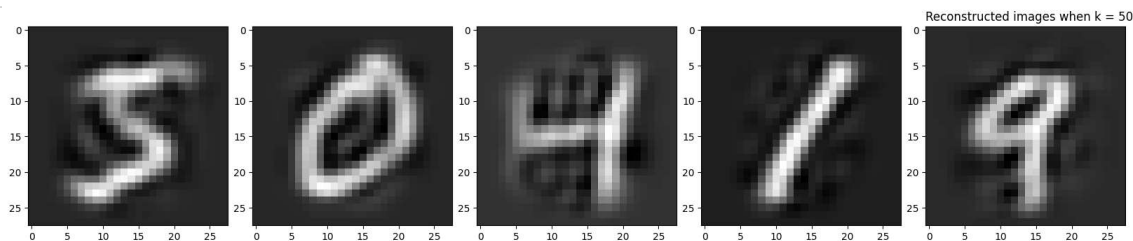
    plt.title(f"Reconstructed images when k = {k}", loc = "left")

def problem_1(data):
    k = 50
    show_first_five_reconstructed(data, k)

def problem_2(data):
    k_list = [10,30,50,100,300,500]
    for k in k_list:
        show_first_five_reconstructed(data, k)

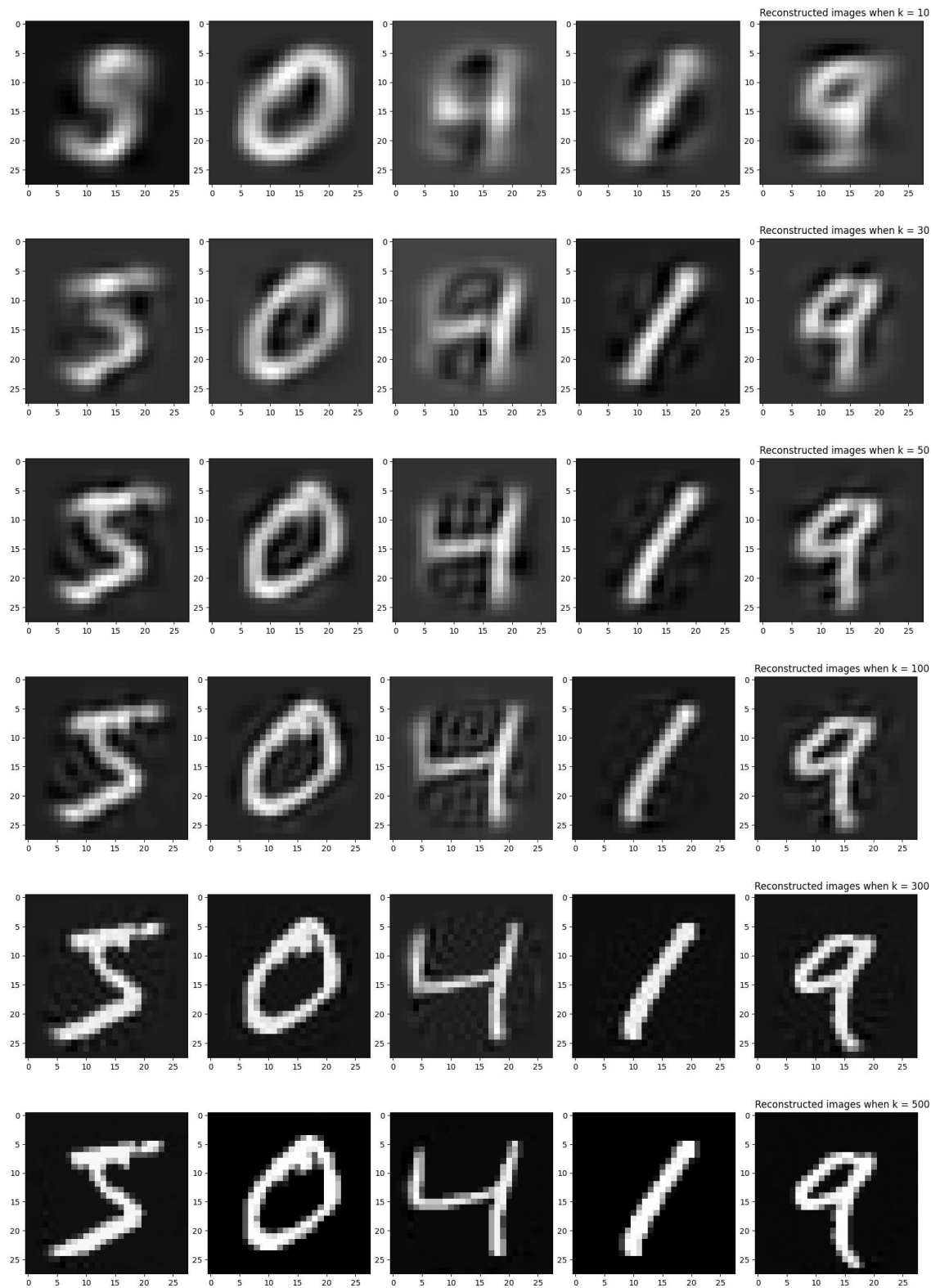
# xx is (70000, 784)
problem_1(xx) # We pick 50, when we could recognize the reconstructed digits by raw eyes, which

```



In [11]:

```
problem_2(xx) # We find that 30 is minimum number of eigenvectors that are needed to reconstruct
```



In []:

Problem 7.5 (\star). Let $A \in \mathbb{R}^{n \times m}$ a rectangular matrix. Show that there exists an orthonormal basis u_1, \dots, u_m of \mathbb{R}^m such that Au_1, \dots, Au_m is an orthogonal family (its vectors are pairwise orthogonal but not necessarily of norm one). (Hint: apply the spectral theorem to $A^\top A$.)

Assume $U = [\vec{u}_1 \ \vec{u}_2 \ \dots \ \vec{u}_m] \in \mathbb{R}^{m \times m}$ is the orthonormal basis that satisfies the problem statement. Then we must have $(AU)^\top AU = U^\top A^\top AU = D$, where D is a diagonal matrix. This has to hold since if AU is orthogonal family, $(A\vec{u}_i)^\top A\vec{u}_j = 0, \forall i \neq j$ and $(A\vec{u}_i)^\top A\vec{u}_i = \|A\vec{u}_i\|_2^2$, which is some non-zero value and is different across i . Now since we should have $U^\top A^\top AU = D$, we rearrange it and could get $A^\top A = UDU^\top$, which is just the spectrum decomposition of matrix $A^\top A$. Thus there exists the orthonormal basis U , which is just the eigenvectors of $A^\top A$ such that $A\vec{u}_i$'s is an orthogonal family.