In [21]:

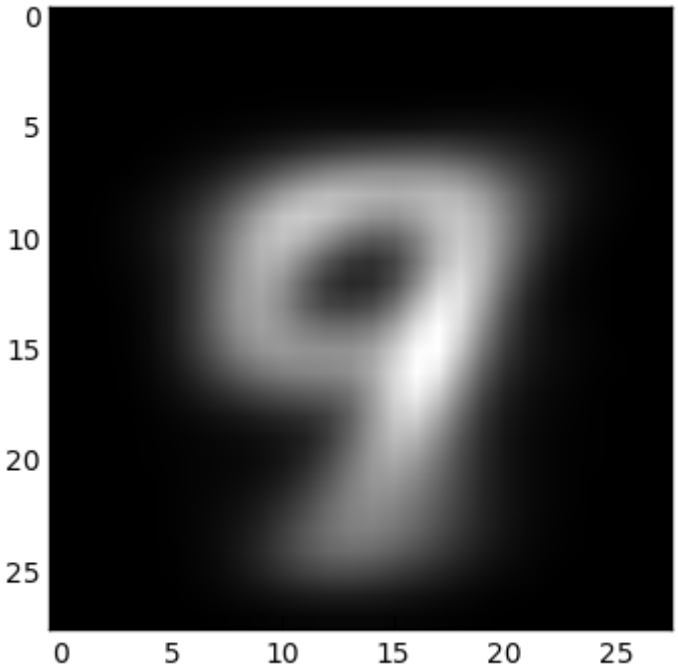
```
import numpy as np
import scipy.stats
import scipy.io
import sklearn.metrics as metrics
import pdb
import pandas as pd
import sys
import csv
import math
import matplotlib.pyplot as plt
import random
import scipy.spatial


%matplotlib inline

# np.set_printoptions(threshold=10)

### preprocessing ###
def load_dataset(filename):
    mat = scipy.io.loadmat(filename)
    return mat['images']

def flatten(mat):
    data = np.zeros((mat.shape[2], mat.shape[0]*mat.shape[1]))
    for i in range(mat.shape[2]):
        data[i] = mat[:,:,i].flatten()
    return data

def split_dataset(data, prop=0.5):
    size = int(data.shape[0]*prop)
    return data[:size], data[size:]


def show_image(X):
    im = X.reshape(28, 28)*255 #Image.fromarray(X[i].reshape(28, 28)*25
5)
    plt.gray()
    plt.imshow(im)
    plt.show()

def kmeans(X, k=5, num_iter=20, init='kmeans++'):
    labels = np.zeros(X.shape[0], dtype=int)
    if init == 'kmeans++':
        centroids = kmeans_plus_init(X, k)
    else:
        centroids = lloyd_init(X, k)
    for i in range(num_iter):
        print("iteration" + str(i))
        labels = update_labels(X, centroids)
        new_centroids = update_centroids(X, labels, k)
        if np.array_equal(new_centroids, centroids):
            break
        centroids = new_centroids
    return labels, centroids

def lloyd_init(X, k):
```

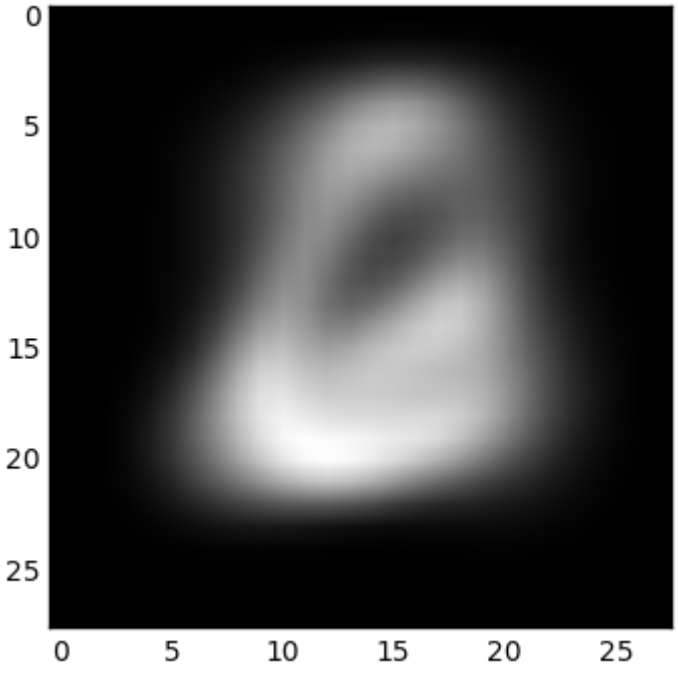
```
    return X[np.random.choice(X.shape[0], k)]

def kmeans_plus_init(X, k):
    first_centroid = X[np.random.choice(X.shape[0], 1)]
    centroids = first_centroid
    for i in range(k):
        sqdists = scipy.spatial.distance.cdist(centroids, X, 'sqeuclidea
n')
        mins = np.argmin(sqdists, axis=0)

        prob = np.empty(sqdists.shape[1])
        for pt in range(sqdists.shape[1]):
            prob[pt] = sqdists[:,pt][mins[pt]]
        prob /= np.sum(prob)

        new_centroid = X[np.random.choice(X.shape[0], 1, p=prob)]
        centroids = np.r_[centroids, new_centroid]
    return centroids

def update_labels(X, centroids):
    sqdists = scipy.spatial.distance.cdist(centroids, X, 'sqeuclidean')
    return np.argmin(sqdists, axis=0)

def update_centroids(X, labels, k):
    centroids = np.empty((k, X.shape[1]))
    for i in range(k):
        centroids[i] = np.mean(X[labels == i], axis=0)
    return centroids
```

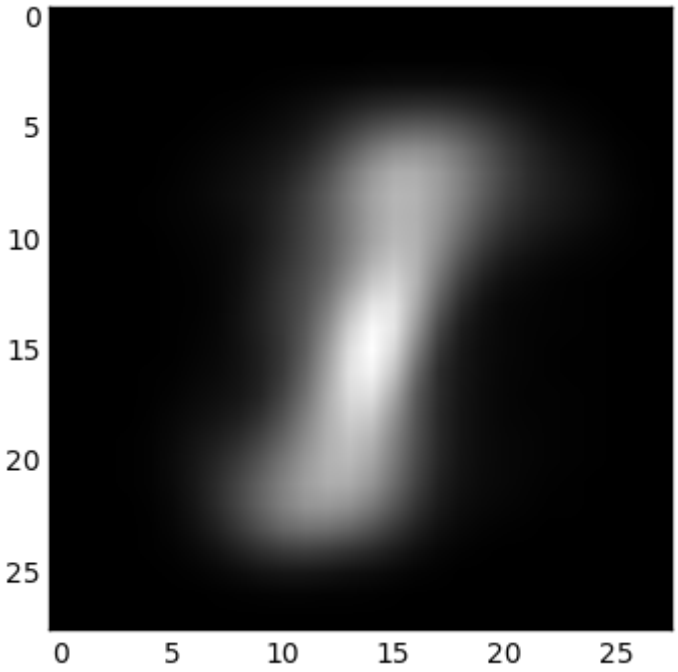
In [ ]:
```
mat = load_dataset('mnist_data/images')
# reshape
X = flatten(mat)
# shuffle
np.random.shuffle(X)

labels_5, c_5 = kmeans(X, init='kmeans++', k=5)
labels_10, c_10 = kmeans(X, init='kmeans++', k=10)
labels_20, c_20 = kmeans(X, init='kmeans++', k=20)
```

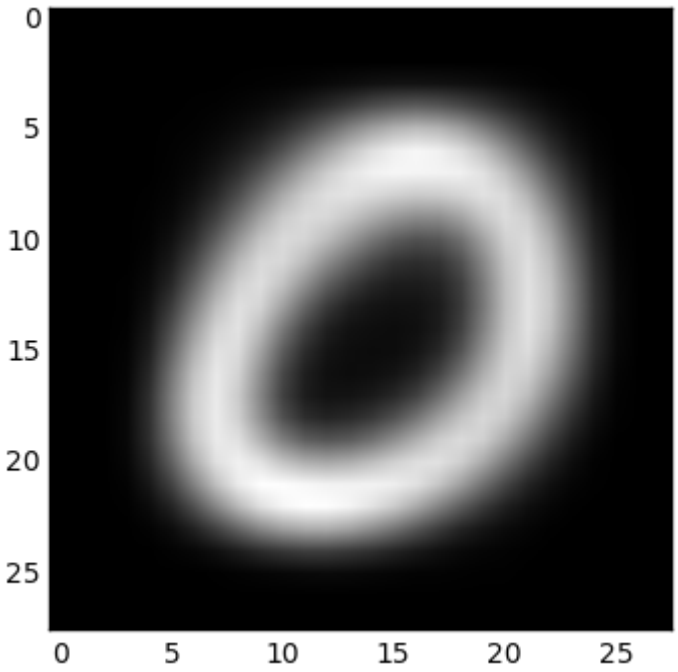
In [3]:
```
c_5 = np.load("centroids_5.npy")
c_10 = np.load("centroids_10.npy")
c_20 = np.load("centroids_20.npy")
```

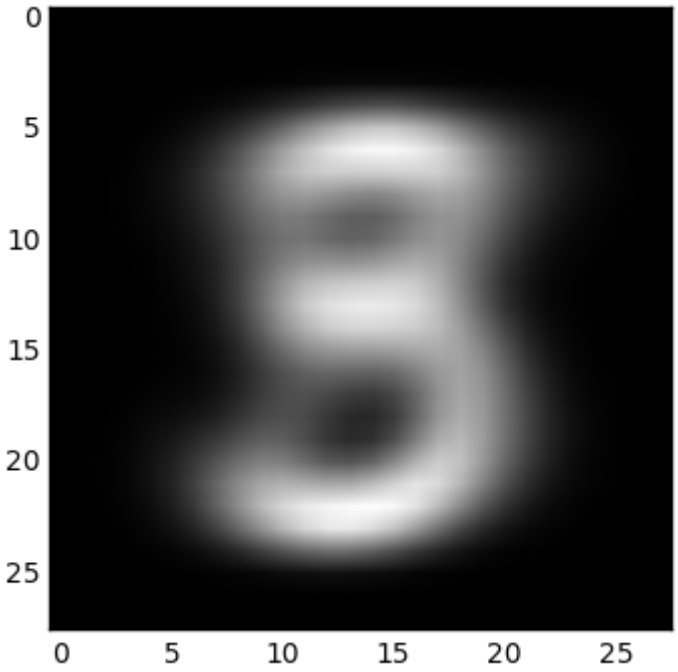
In [26]:
```
for i in range(len(c_5)):
    show_image(c_5[i])
```


0
5
10
15
20
25
0
5
10
15
20
25

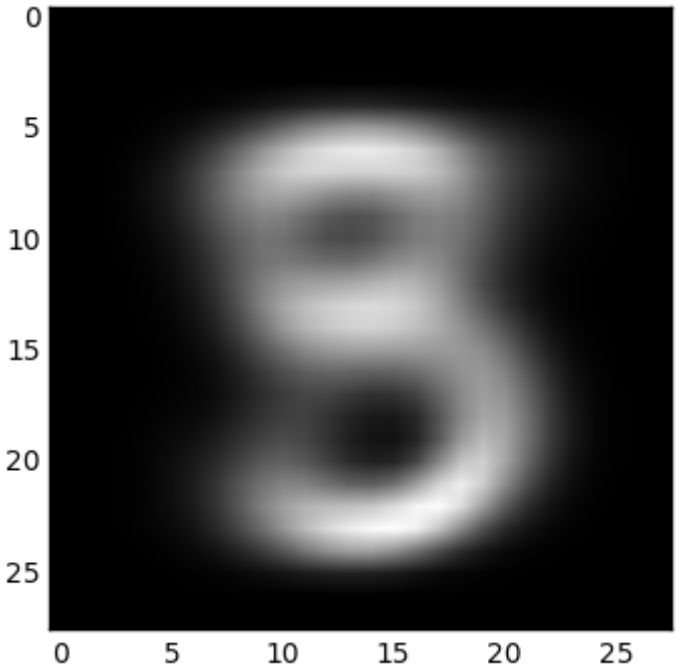

0
5
10
15
20
25
0
5
10
15
20
25

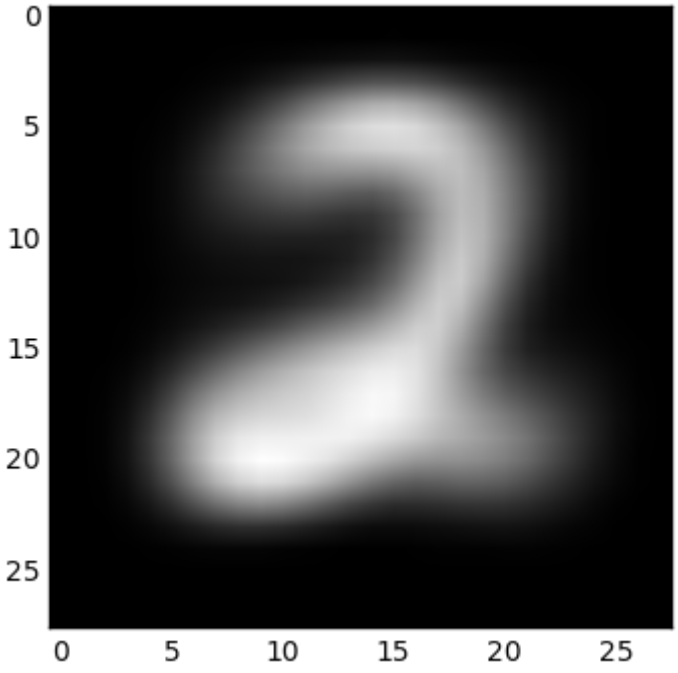

0
5
10
15
20
25
0
5
10
15
20
25

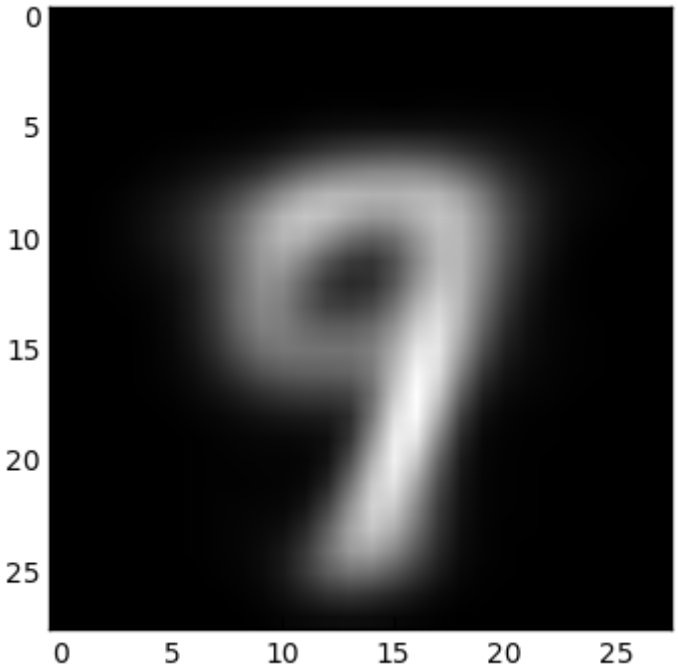

0
5
10
15
20
25
0
5
10
15
20
25

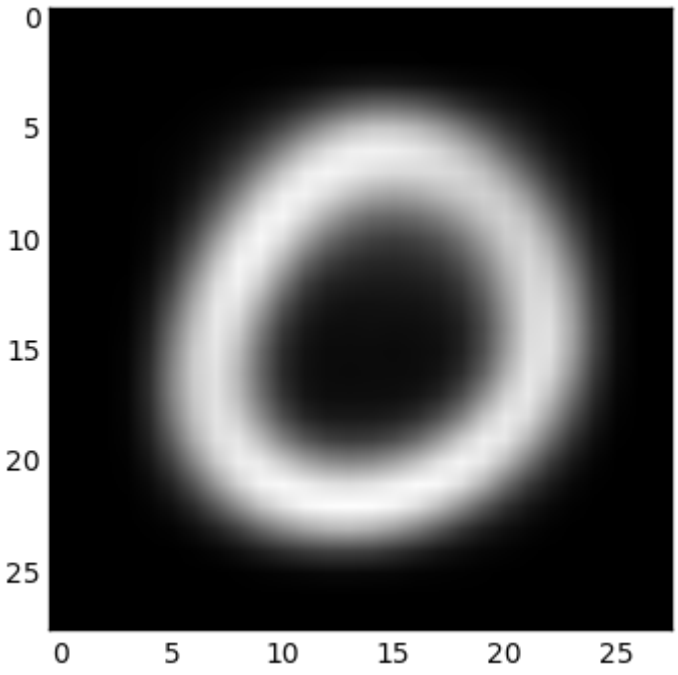

0
5
10
15
20
25
0
5
10
15
20
25

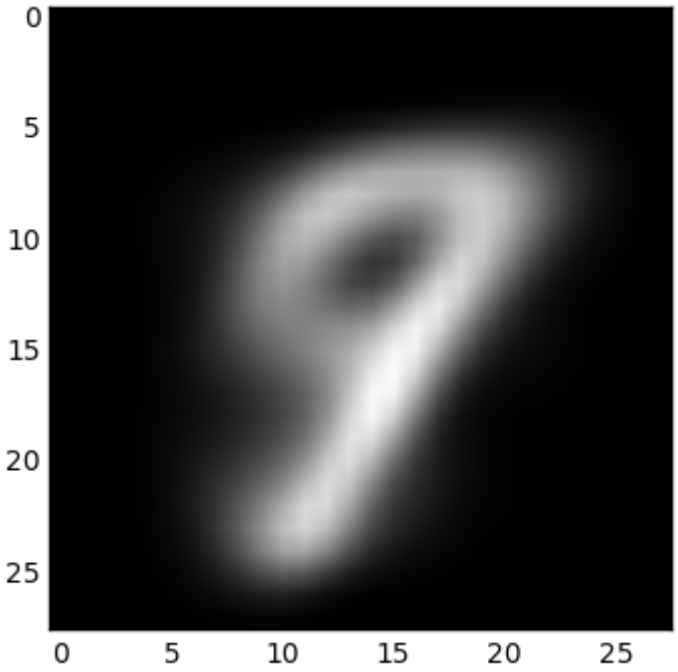
In [27]:
```
for i in range(len(c_10)):
    show_image(c_10[i])
```

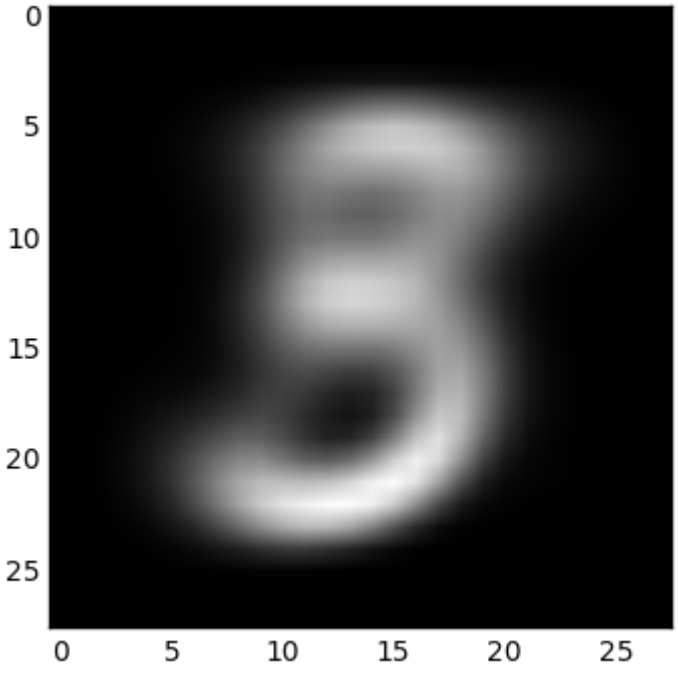

0
5
10
15
20
25
0
5
10
15
20
25

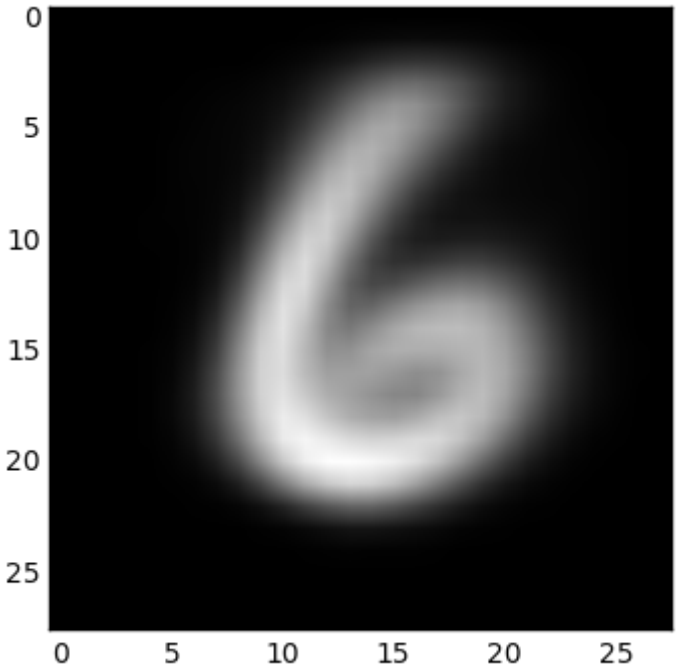

0
5
10
15
20
25
0
5
10
15
20
25

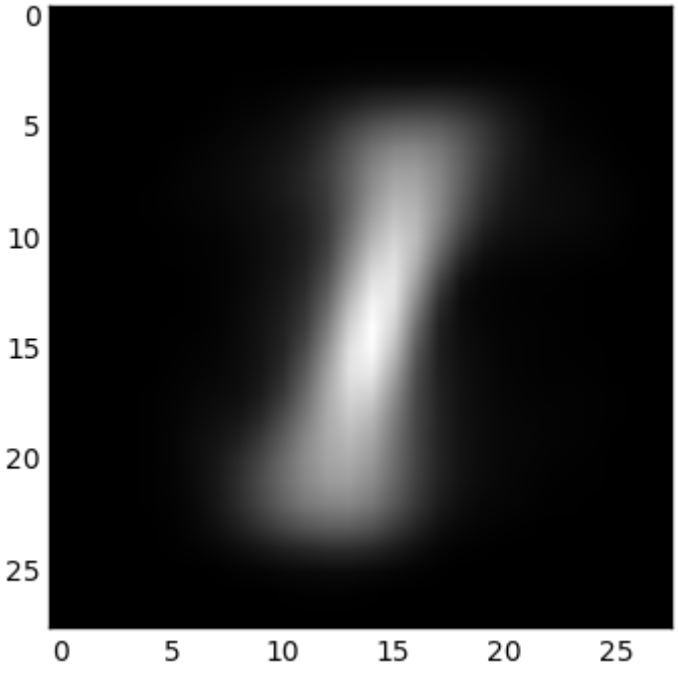

0
5
10
15
20
25
0
5
10
15
20
25

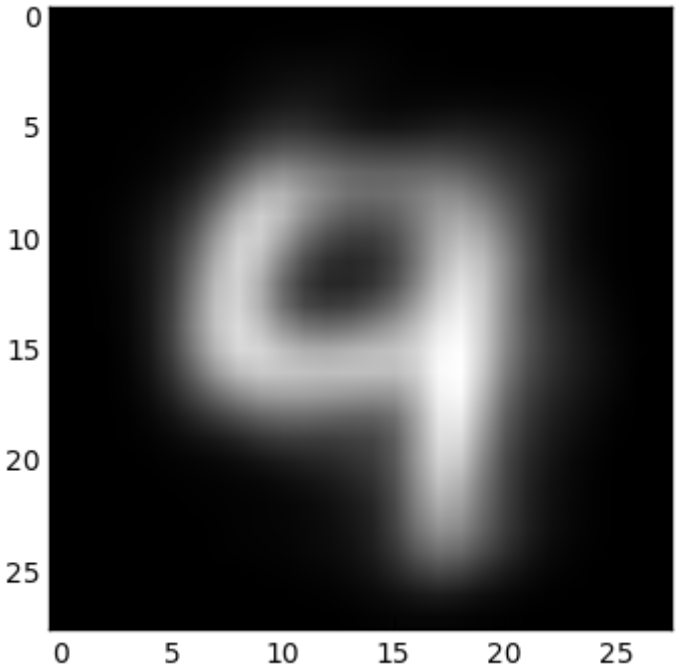

0
5
10
15
20
25
0
5
10
15
20
25

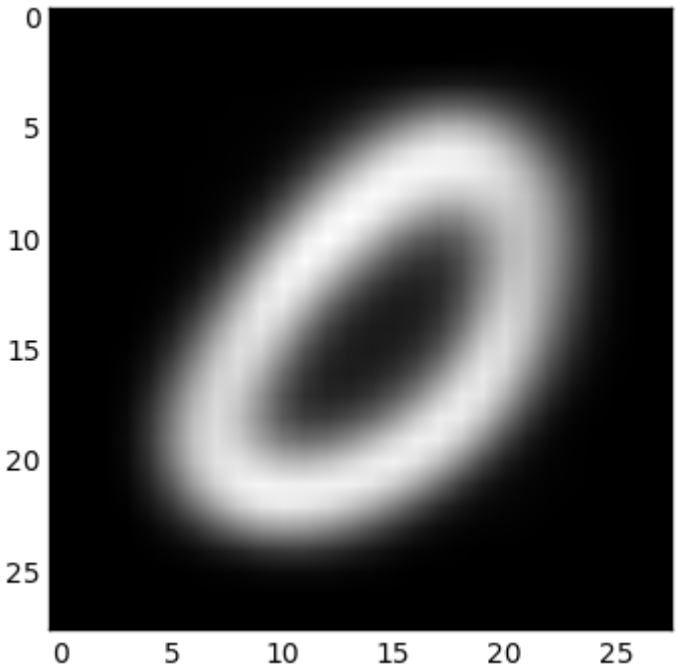

0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25

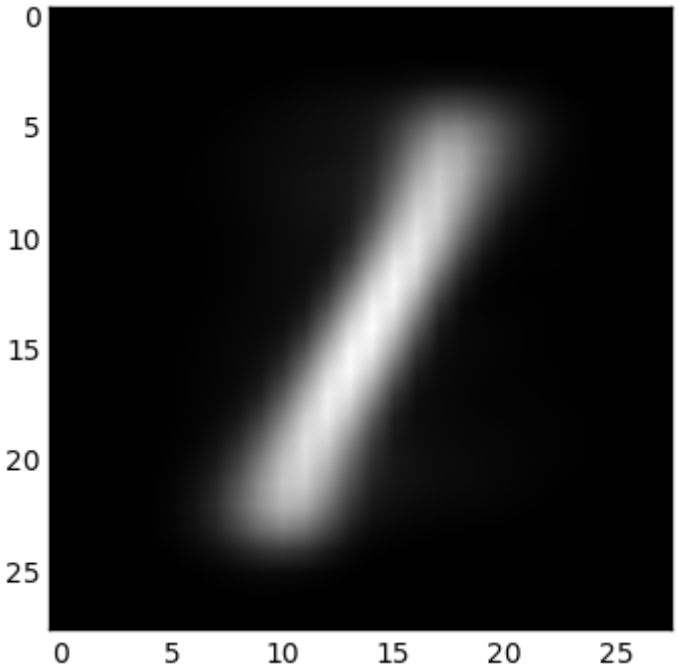

0
5
10
15
20
25
0
5
10
15
20
25

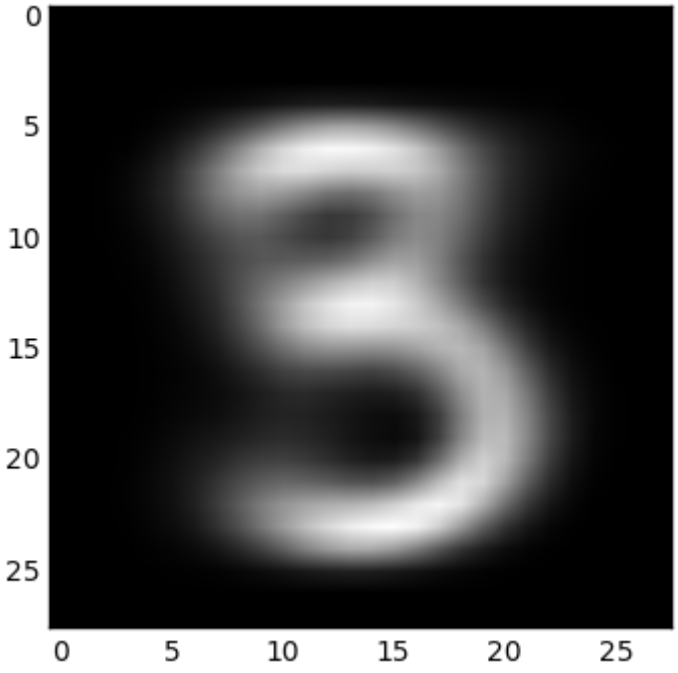

0
5
10
15
20
25
0
5
10
15
20
25

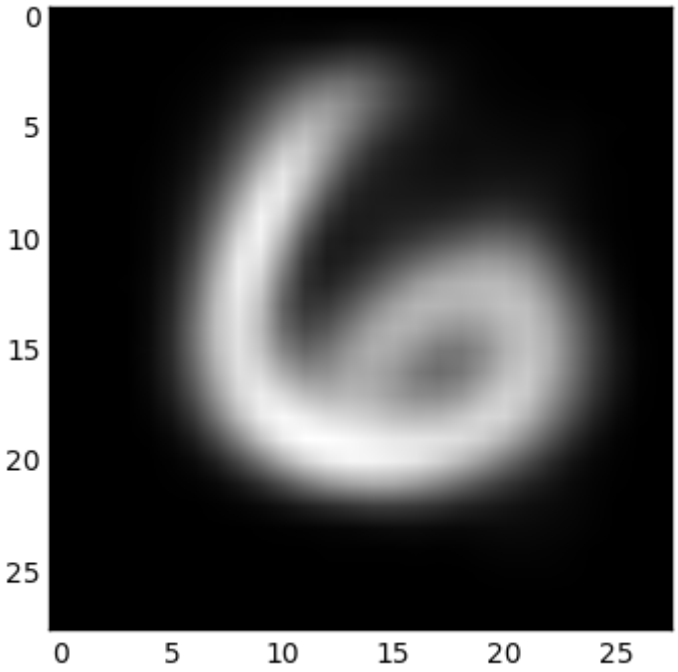

0
5
10
15
20
25
0
5
10
15
20
25

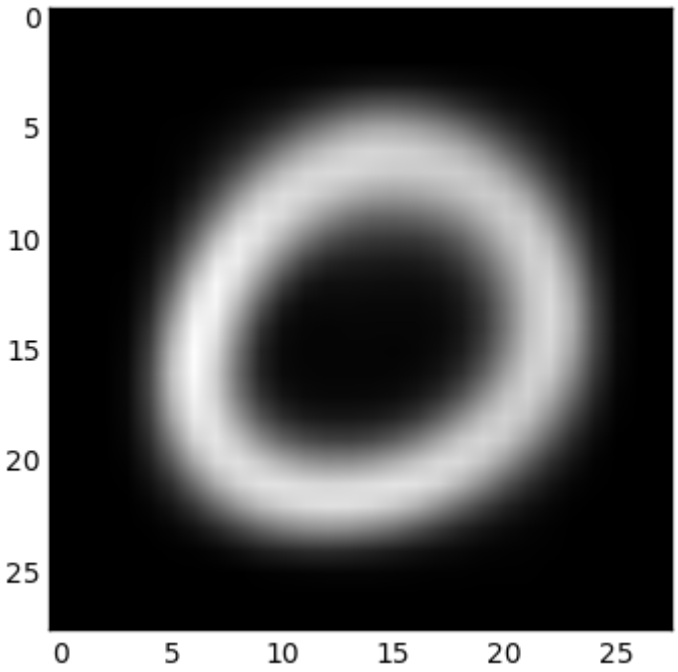

0
5
10
15
20
25
0
5
10
15
20
25

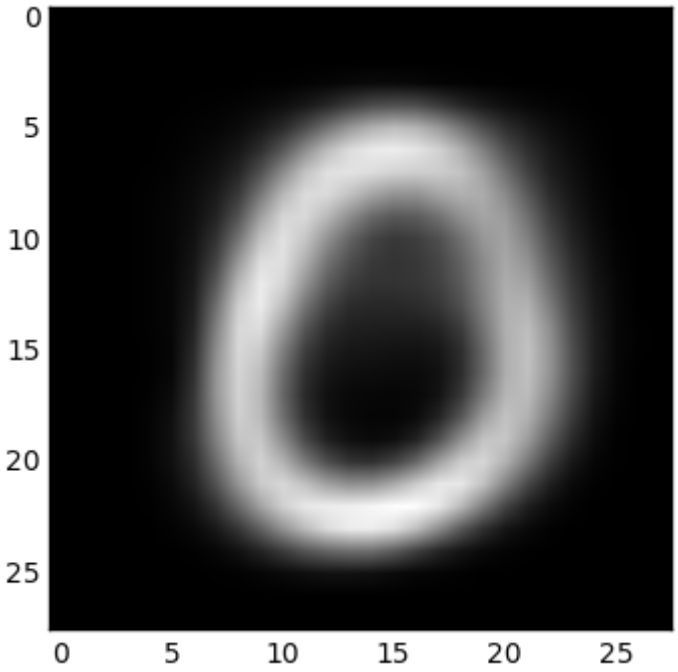
In [28]:
```
for i in range(len(c_20)):
    show_image(c_20[i])
```

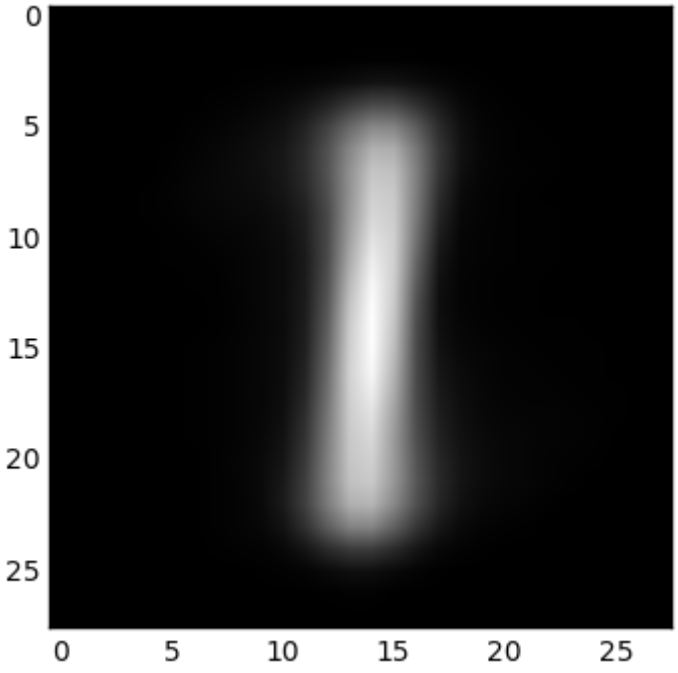

0
5
10
15
20
25
0
5
10
15
20
25

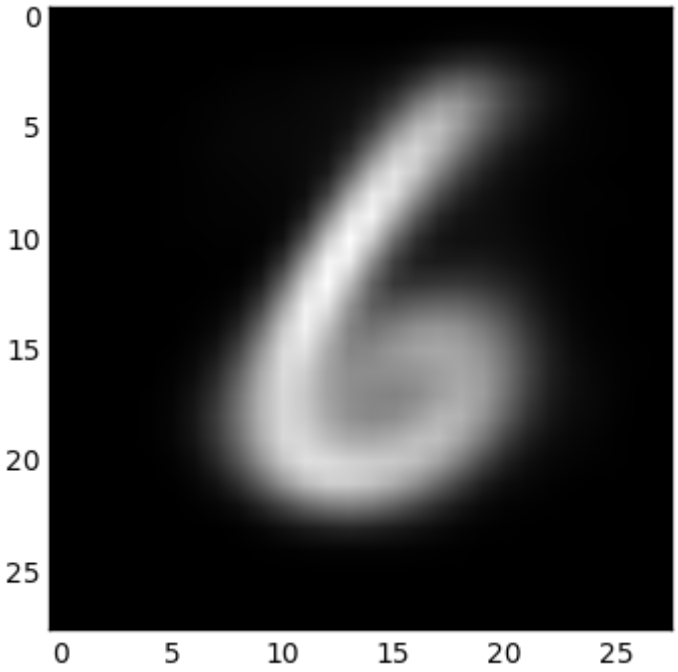

0
5
10
15
20
25
0
5
10
15
20
25

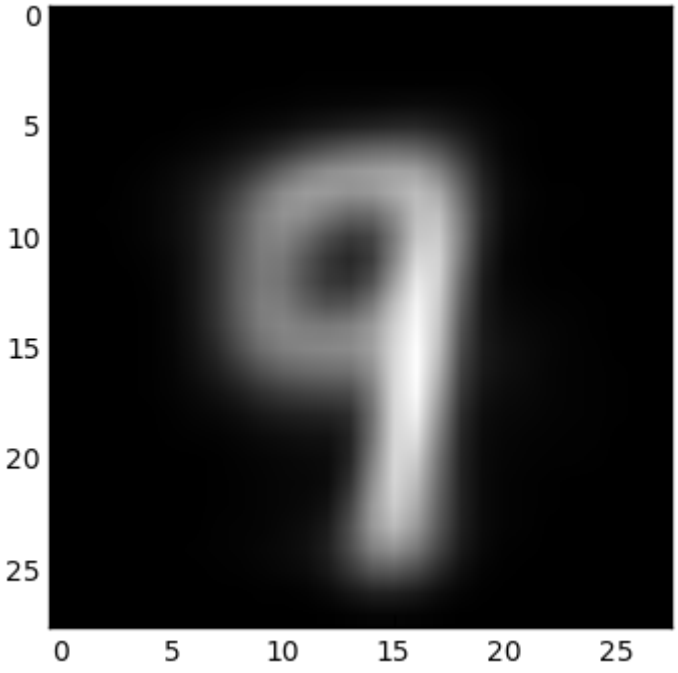

0
5
10
15
20
25
0
5
10
15
20
25

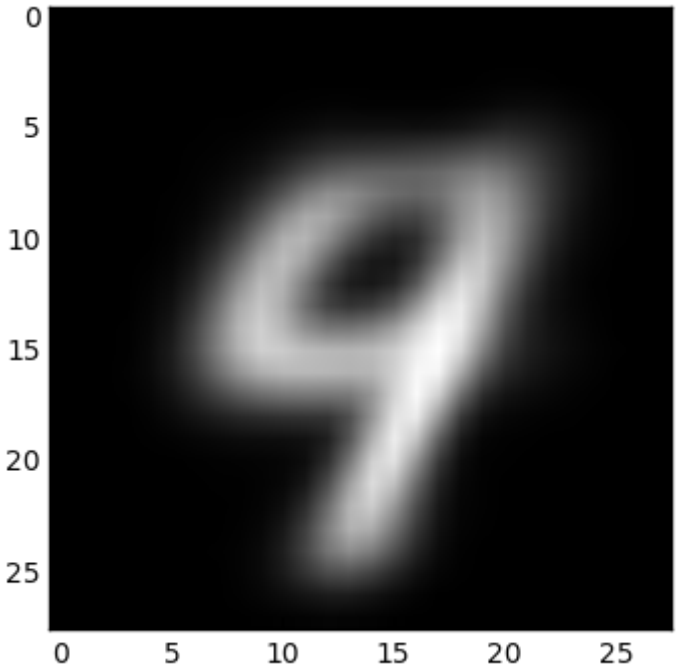

0
5
10
15
20
25
0
5
10
15
20
25

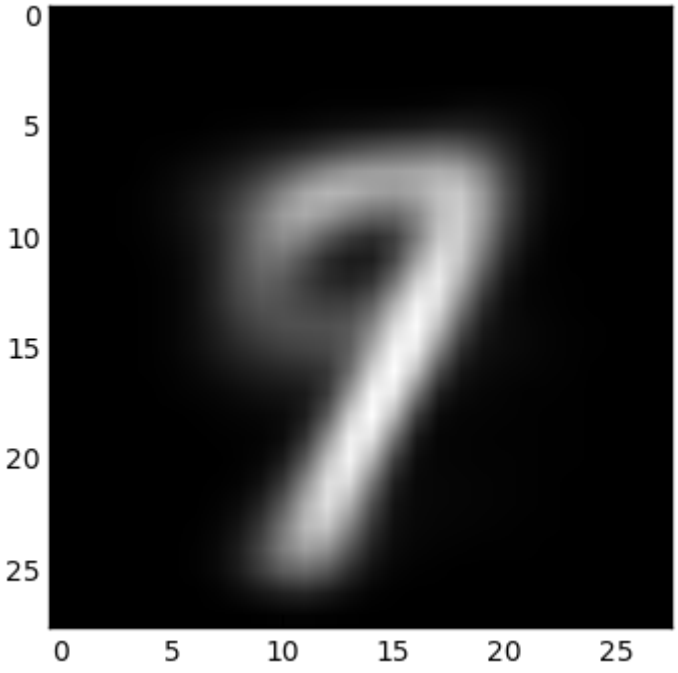

0
5
10
15
20
25
0
5
10
15
20
25

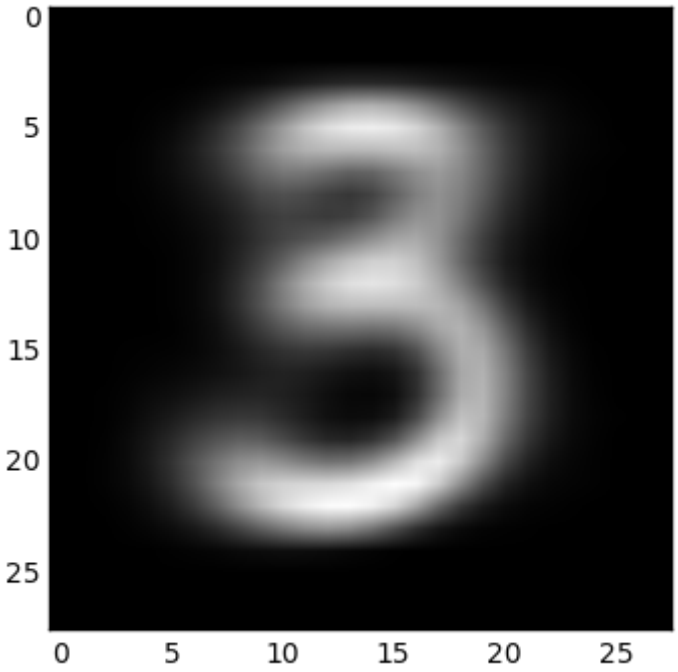

0
5
10
15
20
25
0
5
10
15
20
25

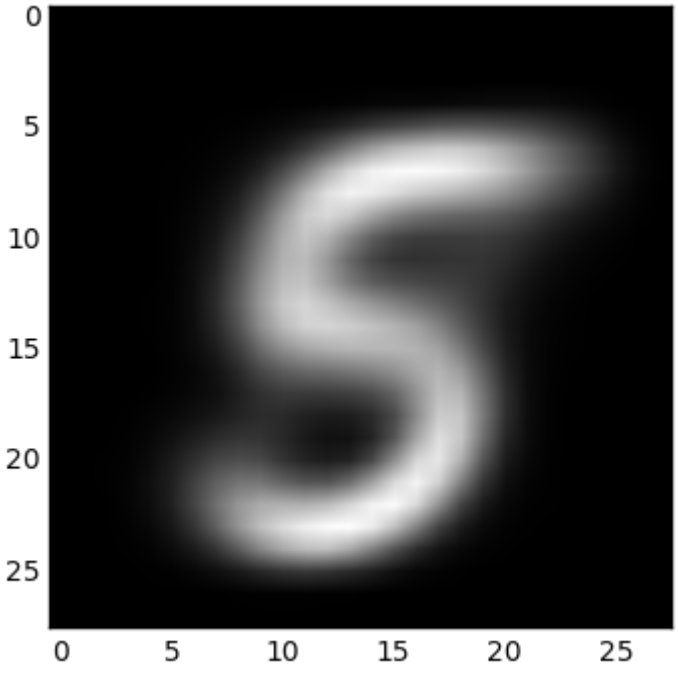

0
5
10
15
20
25
0
5
10
15
20
25

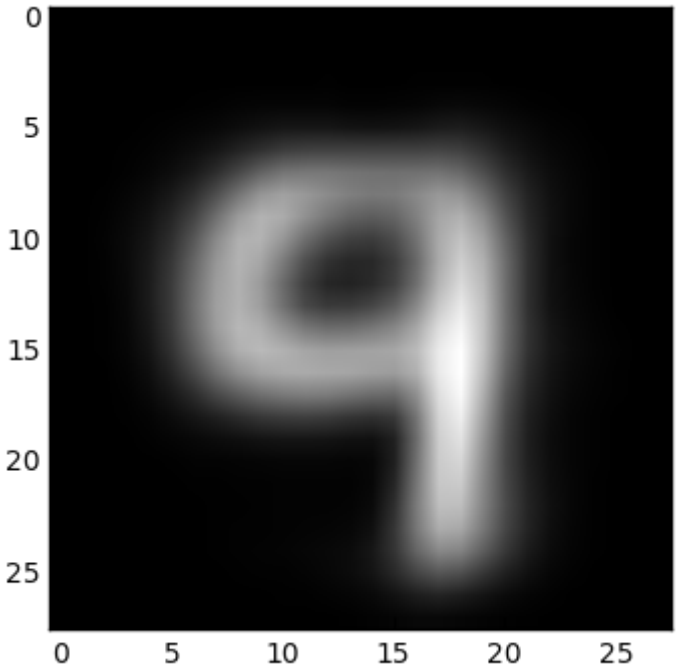

0
5
10
15
20
25
0
5
10
15
20
25

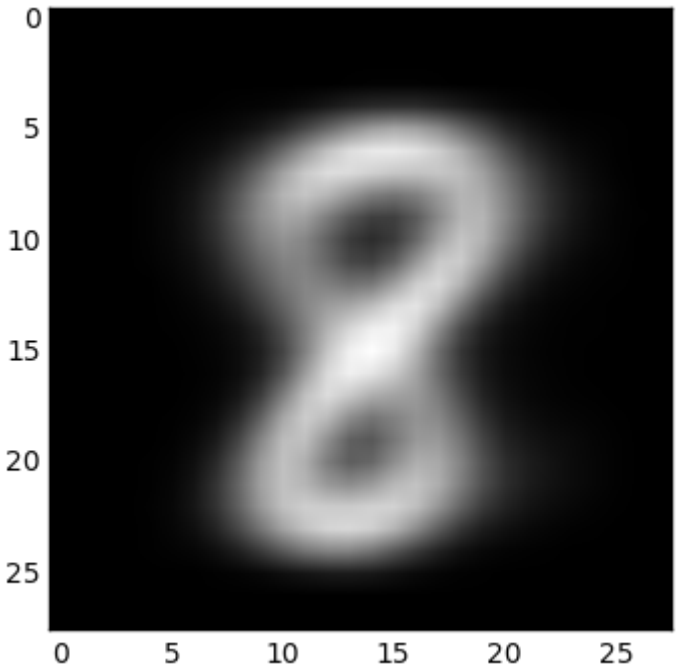

0
5
10
15
20
25
0
5
10
15
20
25

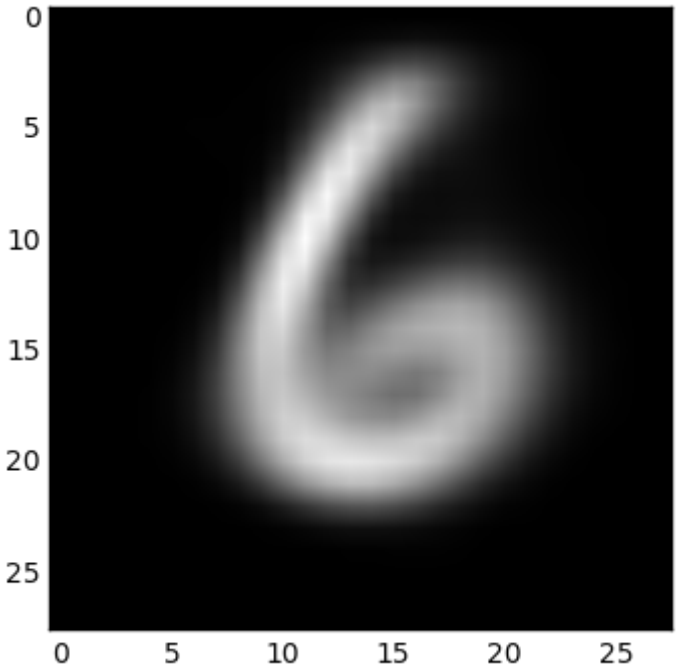

0
5
10
15
20
25
0
5
10
15
20
25

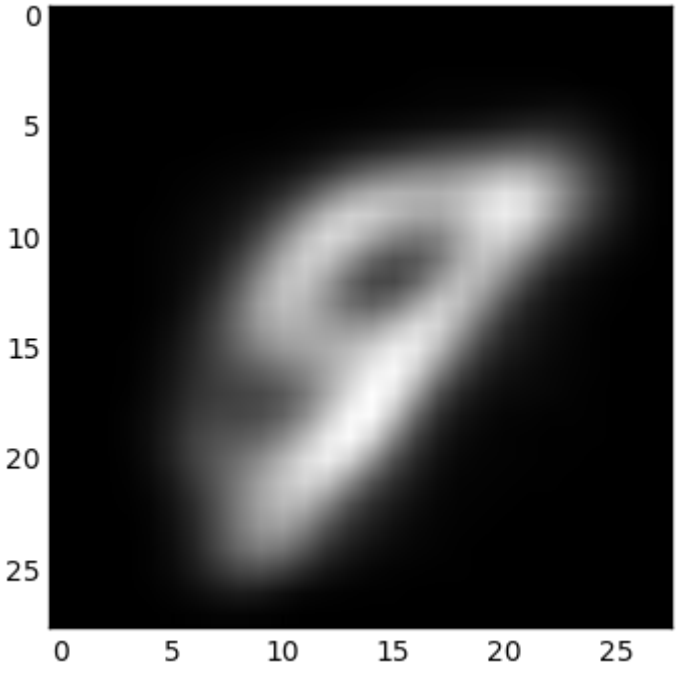

0
5
10
15
20
25
0
5
10
15
20
25

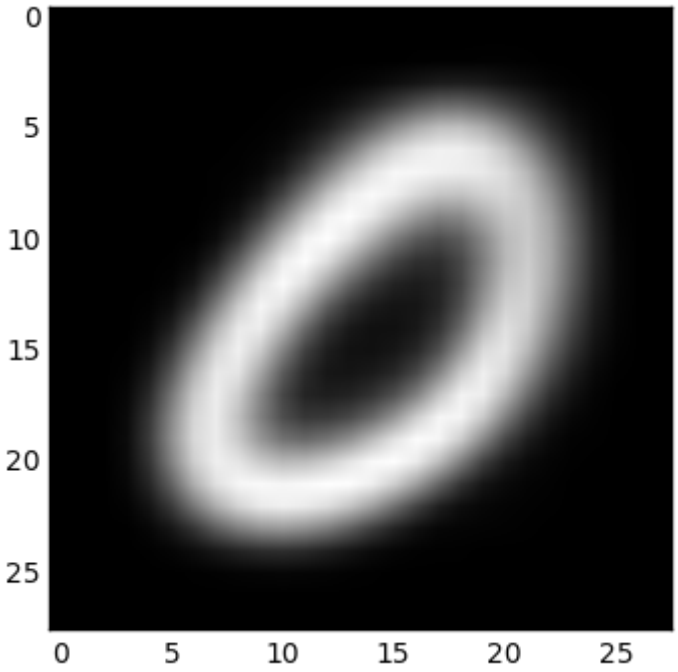

0
5
10
15
20
25
0
5
10
15
20
25

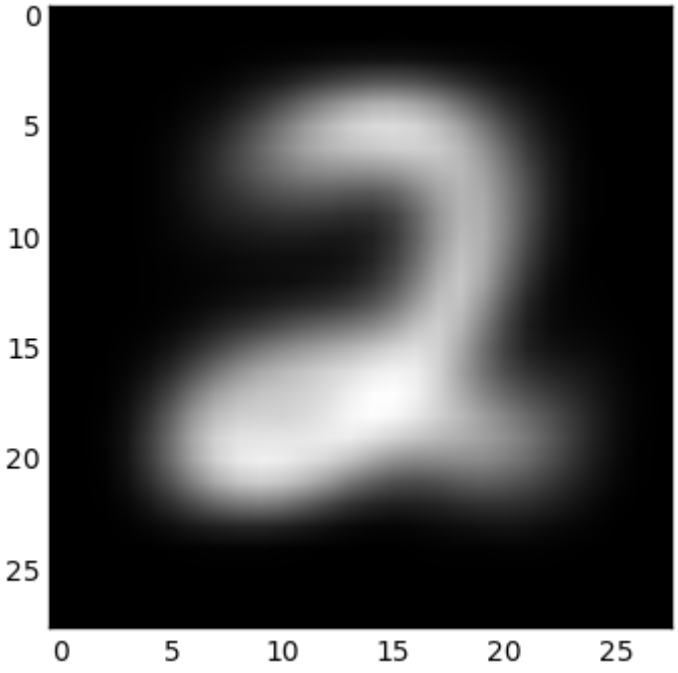

0
5
10
15
20
25
0
5
10
15
20
25

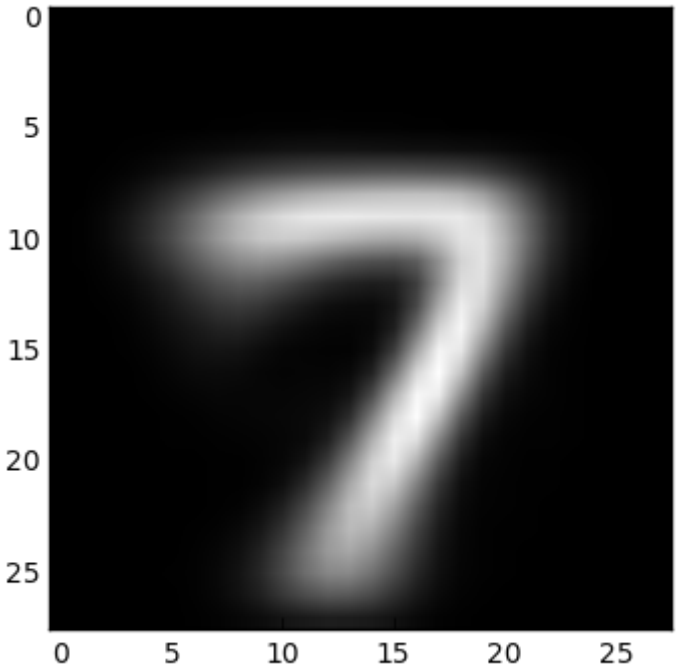

0
5
10
15
20
25
0
5
10
15
20
25

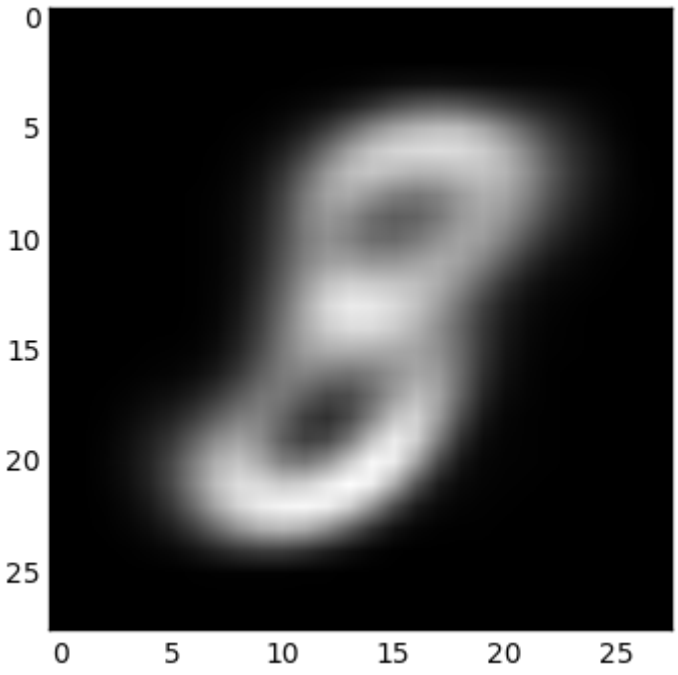

0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25


0
5
10
15
20
25
0
5
10
15
20
25





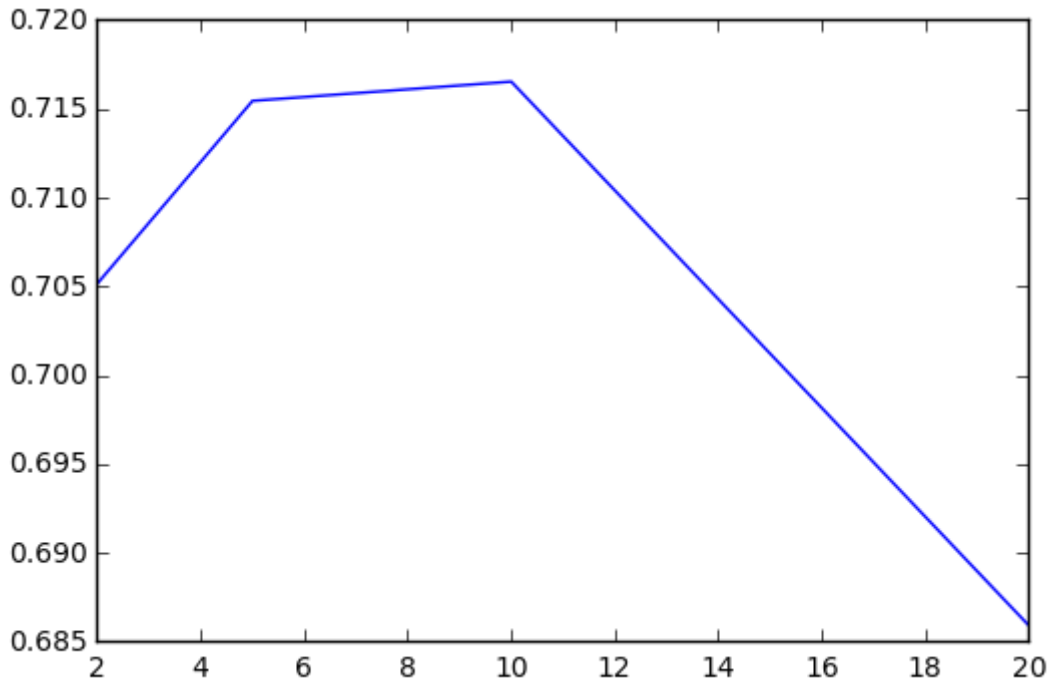
Number 4

In [ ]:
```
import scipy.io
import scipy.sparse
import scipy.linalg
import numpy as np
import sklearn.metrics as metrics
import pdb
import matplotlib.pyplot as plt
import csv

%matplotlib inline

def svd(X, k):
    X_nan = np.nan_to_num(X)
    u,s,v = scipy.sparse.linalg.svds(X_nan, k)
    return u.dot(np.diag(s)).dot(v)

def mse(X, model):
    error = 0
    for i in range(X.shape[0]):
        for j in range(X.shape[1]):
            if not np.isnan(X[i][j]):
                error += (model[i][j]-X[i][j])**2
    return error

def predict(X_test, model):
    pred = np.empty(X_test.shape[0])
    for i, row in enumerate(X_test):
        user, joke = row-1
        mean = np.mean(model[user])
        if model[user][joke] >= 0:
            pred[i] = 1
        else:
            pred[i] = 0
    return pred


def train(X_train, k=5, reg=100, threshold=1e-4, num_iter=1000):
    R = np.nan_to_num(X_train)
    U = np.random.rand(k, R.shape[0])
    V = np.random.rand(k, R.shape[1])
    for x in range(num_iter):
        old_U, old_V = U, V
        U = scipy.linalg.solve(V.dot(V.T) + reg*np.eye(k), V.dot(R.T))
        V = scipy.linalg.solve(U.dot(U.T) + reg*np.eye(k), U.dot(R))
        diff_U = np.linalg.norm(old_U-U)
        diff_V = np.linalg.norm(old_V-V)
        print("U diff is: " + str(diff_U))
        print("V diff is: " + str(diff_V))
        if diff_U < threshold or diff_V < threshold:
            break
        # U, V = new_U, new_V
    print("iteration: " + str(x))
    return U.T.dot(V)
```

In [ ]:
```
X = scipy.io.loadmat('joke_data/joke_train')
X_train = X['train']
X_validate = np.loadtxt('joke_data/validation.txt', delimiter=',', dtype=
)
X_test = np.loadtxt('joke_data/query.txt', dtype=int, delimiter=',')
X_test_ids, X_test = X_test[:,0], X_test[:,1:]
X_validate, labels_validate = X_validate[:,:2], X_validate[:,2]
dimensions = [2, 5, 10, 20]

svd_accuracies = []
for k in dimensions:
    model = svd(X_train, k)
    error = mse(X_train, model)
    print(error)
    pred_labels = predict(X_validate, model)
    accuracy = metrics.accuracy_score(labels_validate, pred_labels)
    print("Validation accuracy: {0}".format(accuracy))
    svd_accuracies.append(accuracy)


closed_accuracies = []
for k in dimensions:
    model = train(np.sign(X_train), k)
    error = mse(X_train, model)
    print(error)
    pred_labels_closed = predict(X_validate, model)
    accuracy = metrics.accuracy_score(labels_validate, pred_labels_close
d)
    print("Validation accuracy: {0}".format(accuracy))
    closed_accuracies.append(accuracy)

model = train(np.sign(X_train), 20)
pred_labels_test = predict(X_test, model)

c = csv.writer(open("kaggle.csv", "wt"))
c.writerow(['Id', 'Category'])
for i in range(len(pred)):
  c.writerow((i+1, int(pred_labels_test[i])))
```

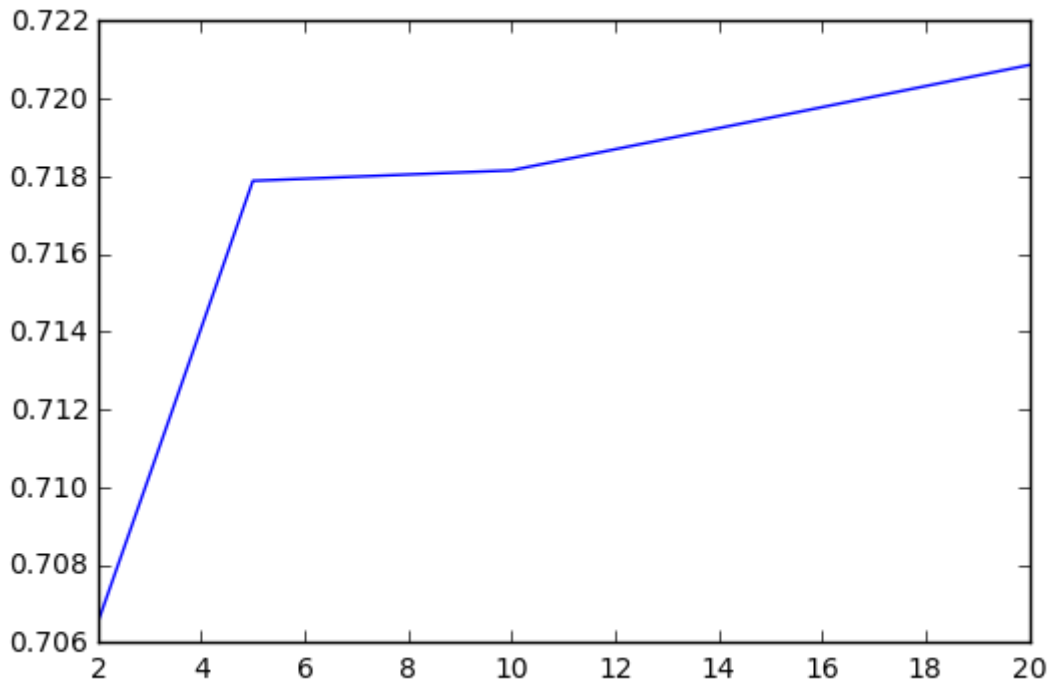
In [32]:
```
print(svd_accuracies)
plt.plot(dimensions, svd_accuracies)
plt.show()
```

```
[ 0.70514905  0.71544715  0.71653117  0.68590786]
```



In [33]:
```
print(closed_accuracies)
plt.plot(dimensions, closed_accuracies)
plt.show()
```

```
[ 0.70650407  0.71788618  0.71815718  0.72086721]
```



In [ ]:

2. PRINCIPAL COMPONENT ANALYSIS

$$A_r \triangleq \min_{M \in \mathbb{R}^{m \times n}} \|A - M\|_F \;:\; \text{rank}(M) \leq r$$

where A, M are p.s.d.

a) Show that $A_1 = mm^T$

where $m \triangleq \min_{m \in \mathbb{R}^n} \|A - mm^T\|_F$

Prove: $\min_{m \in \mathbb{R}^n} \|A - mm^T\|_F = \min_{M \in \mathbb{R}^{n \times n}} \|A - M\| \;:\; \text{rank}(M) \leq 1$

In order to do this, I must prove two facts about mm^T

1. mm^T is positive semidefinite
2. mm^T has rank ≤ 1

Then, the minimizations will be equivalent

(1) $mm^T \succeq 0$

$$x^T mm^T x = (m^T x)^2 \geq 0 \quad \forall x \Rightarrow mm^T \succeq 0$$

(2) $\text{rank}(mm^T) \leq 1$

$$mm^T = \begin{bmatrix} | & | & & | \\ m_1 \vec{m} & m_2 \vec{m} & \cdots & m_n \vec{m} \\ | & | & & | \end{bmatrix}$$

$\Rightarrow$ columns of mm^T are all multiples of $\vec{m}$

thus there is at most 1 linearly independent column

so $\text{rank}(mm^T) \leq 1$

b) $f: \mathbb{R}^n \to \mathbb{R}$ s.t. $f(m) = \|A - mm^T\|_F^2$

$\nabla f(m) = ?$

$\|A - mm^T\|_F^2 = Tr\left((A - mm^T)^T(A - mm^T)\right)$

$= Tr\left((A^T - mm^T)(A - mm^T)\right)$

$= Tr\left(A^TA - mm^TA - A^Tmm^T - mm^Tmm^T\right)$

$= Tr(A^TA) - Tr(mm^TA) - Tr\left((mm^TA)^T\right) - Tr(m^Tmm^Tm)$

$= Tr(A^TA) - 2\,Tr(mm^TA) - (m^Tm)^2$

$f = Tr(A^TA) - 2m^TAm - (m^Tm)^2$

$\nabla_m f = 0 - 4m^TA - 2(m^Tm)\cdot 2m^T$

$= \underline{4m^TA - 4m^Tmm^T}$

c). $0 = 4m^TA_1 - 4m^Tmm^T$

$m^TA_1 = m^Tmm^T$

$A_1m = \underline{mm^T}m$

Define some unit vector $\vec{v}$ s.t. $\vec{v}^T\vec{v} = 1$

$\vec{m} = c\vec{v}$ where $c \in \mathbb{R}$, thus $\vec{m}$ is a scaling of $\vec{v}$

$A_1c\vec{v} = c\vec{v}\,c\vec{v}^Tc\vec{v}$

$A\vec{v} = c^2\vec{v}\vec{v}^T\vec{v}$

$A\vec{v} = c^2\vec{v}$ b/c $\vec{v}^T\vec{v} = 1$

This can only be true if

$c^2 = \lambda$ and $\vec{v}$ is the corresponding eigenvector

$c = \sqrt{\lambda}$

Thus, $A_1 = \sqrt{\lambda}\vec{v}$

2) $D \triangleq \{m : m = \sqrt{\lambda_k}\,\vec{v}_k \;\; \forall \text{ eigenvalue } \lambda_k, \text{ eigenvector } \vec{v}_k\}$

Prove $\min \|A - mm^T\|_F \Rightarrow A_1 = \lambda_1 \vec{v}_1 \vec{v}_1^T$

$m \in D$ Hint: $m = \sqrt{\lambda_1}\,\vec{v}_1$

$f = \|A - mm^T\|_F$

$f^2 = \mathrm{Tr}\left((A - mm^T)^T (A - mm^T)\right)$

$= \mathrm{Tr}(A^TA) - 2\,m^TAm + (m^Tm)^2$

$= \sum_{i=1}^{n} \lambda_i^2 - 2\sqrt{\lambda_k} v^T A \sqrt{\lambda_k} v + (\lambda_k v^T v)^2$

$= \sum_{i=1}^{n} \lambda_i^2 - 2\lambda_k v^T \lambda_k v + \lambda_k^2$

$= \sum_{i=1}^{n} \lambda_i^2 - 2\lambda_k^2 + \lambda_k^2$

$= \sum_{i=1}^{n} \lambda_i^2 - 3\lambda_k^2$

$A_1 = \min_{k \le r} \sum_{i=1}^{n} \lambda_i^2 - 3\lambda_k^2$

Since $\lambda_1 > \lambda_2 > \dots > \lambda_r$

$\Rightarrow \lambda_1$ would minimize f^2

$\Rightarrow A_1 = \lambda_1 v_1 v_1^T$ ✓

3 Singular Value Decomposition

a) Unitarily Invariant ∈ Given $A \in \mathbb{R}^{m \times n}$ show

$$\sigma(A) = \sigma(Q_1 A) = \sigma(A Q_2)$$

where $\forall Q_1 \in \mathbb{R}^{m \times m}$ orthonormal $\quad \sigma(\cdot) \triangleq$ set of singular values

$\forall Q_2 \in \mathbb{R}^{n \times n}$ orthonormal

$$\begin{bmatrix} — & q_1 & — \\ — & q_2 & — \\ & \vdots & \\ — & q_n & — \end{bmatrix} \begin{bmatrix} | & & | \\ a_1 & \cdots & a_m \\ | & & | \end{bmatrix} = \begin{bmatrix} q_1^T a_1 & & q_1^T a_m \\ & \vdots & \\ q_n^T a_1 & \cdots & q_n^T a_m \end{bmatrix}$$

$A = U \Sigma V^T \rightarrow \sigma(A) = \text{diag}(\Sigma)$ or $A = U \Sigma V^T$ (n×d, d×d, d×d; n×d)

$Q_1 A = Q_1 U \Sigma V^T$

know $Q_1^T Q_1 = I$, & $U^T U = I$

$$(Q_1 U)^T (Q_1 U) = U^T Q_1^T Q_1 U = U^T U = I \quad \checkmark$$

$\sigma(Q_1 A) = \text{diag}(\Sigma)$

& $A Q_2 = U \Sigma V^T Q_2$

orthonormal b/c $(V^T Q_2)^T (V^T Q_2) = I$

$\sigma(A Q_2) = \text{diag}(\Sigma)$

b) show $\|A\|_F = \sqrt{\sum_{i=1}^{r} \sigma_i(A)^2}$

$\sqrt{Tr(AA^T)} = \sqrt{\sum_{i=1}^{r} \sigma_i(A)^2}$

$$Tr(AA^T) = \begin{bmatrix} \sigma_1(A) & \sigma_2(A) & \cdots & \sigma_r(A) \end{bmatrix} \begin{bmatrix} \sigma_1(A) \\ \sigma_2(A) \\ \vdots \\ \sigma_r(A) \end{bmatrix}$$

$Tr(U \Lambda U^T) = (\text{diag}(\Sigma_1))^T (\text{diag}(\Sigma_1))$

$\sum_{i=1}^{r} \lambda_i(AA^T) = \sum_{i=1}^{r} (\sigma_i(A))^2$

$\lambda_i = \sigma_i^2$ ✓

Thus $\|A\|_F \propto \sigma_i(A)$ so since any orthonormal transformation of A has same σ_i, it will also have same Frobenius norm

$$\sqrt{(\sigma_1^2 + \sigma_2^2)} \leq \sqrt{\sigma_1^2} + \sqrt{\sigma_2^2}$$

c). Consider $\forall A$ of $\text{rank}(A) = r$

$$\|A\| \leq \|A\|_F \leq \|A\|_* \leq \sqrt{r}\|A\|_F \leq r\|A\|$$

① $\sigma_i(A) \leq \sqrt{\sum_{i=1}^{r} \sigma_i(A)^2}$

$\sigma_i(A)^2 \geq 0 \quad \forall i$

Thus $\sqrt{\sum_{i=1}^{r} \sigma_i(A)^2} = \sqrt{\sigma_1(A)^2 + \sum_{i=2}^{r} \sigma_i(A)^2} \geq \sqrt{\sigma_1(A)^2}$

② $\sqrt{\sum_{i=1}^{r} \sigma_i(A)^2} \leq \sum_{i=1}^{r} \sigma_i(A)$

$\sqrt{\sigma_1(A)^2 + \sigma_2(A)^2 + \cdots + \sigma_r(A)^2} \leq \sqrt{\sigma_1(A)^2} + \sqrt{\sigma_2(A)^2} + \cdots \sqrt{\sigma_r(A)^2}$

by Triangle Inequality

③ $\sum_{i=1}^{r} \sigma_i(A) \leq \sqrt{r}\sqrt{\sum_{i=1}^{r} \sigma_i(A)^2}$

Have $\|x\|_2 \leq \|x\|_1 \leq \sqrt{r}\|x\|_2$

$\|\text{diag}(\Sigma)\|_2 = \sqrt{\sum_{i=1}^{r} \sigma_i(A)^2}$ $\quad A = U\Sigma V^T$

$\|\text{diag}(\Sigma)\|_1 = \sum_{i=1}^{r} |\sigma_i(A)| = \sum_{i=1}^{r} \sigma_i(A)$ $\quad$ b/c $\sigma_i(A) \geq 0$

Thus $\|\text{diag}(\Sigma)\|_2 \leq \|\text{diag}(\Sigma)\|_1 \leq \sqrt{r}\|\text{diag}(\Sigma)\|_2$

④ $\sqrt{r}\|\text{diag}(\Sigma)\|_2 \leq r\|A\| = r\sigma_1(A)$

$\sqrt{r}\sqrt{\sigma_1(A)^2 + \sum_{i=1}^{r} \sigma_i(A)^2} \leq r\sqrt{\sigma_1(A)^2}$

$\sqrt{r}\sqrt{\sigma_1(A)^2 + \sigma_2(A)^2 + \cdots + \sigma_r(A)^2} \leq \sqrt{r}\sqrt{\underbrace{\sigma_1(A)^2 + \cdots + \sigma_1(A)^2}_{r}}$

$\sigma_1(A)^2 \geq \sigma_i(A)^2 \quad \forall i$ ✓

$\Rightarrow \sqrt{r}\|A\|_F \leq r\|A\|$

Show

a) $\|A\| \leq 1 \iff I - A^T A \succeq 0$

$\iff I - AA^T \succeq 0$

$\iff D = \begin{bmatrix} I & A \\ A^T & I \end{bmatrix} \succeq 0$

$\|A\| \leq 1 \iff \sigma_i(A) \leq 1 \rightarrow \sigma_i(A) \leq 1 \ \forall i$

$A = U \Sigma V^T$

$A^T A = V \Lambda V^T \ \ w/ \ \sigma_i(A) = \sqrt{\lambda_i} \rightarrow \Lambda = \Sigma^2$

$AA^T = V \Lambda V^T \ \ w/ \ \sigma_i(A) = \sqrt{\lambda_i}$

$D \succeq 0$ iff $\lambda_i \geq 0 \ \forall$

$(I - A^T A)x = (1 - \lambda_i)x$

$\lambda_i \leq 1 \ \forall i \Rightarrow 1 - \lambda_i \geq 0 \ \forall i$

$\Rightarrow I - A^T A \succeq 0$

$\Rightarrow I - AA^T \succeq 0$

$(D - \lambda I) = \begin{vmatrix} I - \lambda & A \\ A^T & I - \lambda \end{vmatrix} = (I - \lambda I)^2 - AA^T = 0$

$\lambda^2 - \lambda \cdot Tr(D) + \det(D) = 0$

$\lambda^2 - 2\lambda I + (I - AA^T) = 0$

$\lambda_1 + \lambda_2 = Tr(D) = 2I$

$\lambda_1 \lambda_2 = \det(D) = I - AA^T \succeq 0$

$\Rightarrow \lambda_1, \lambda_2 > 0 \Rightarrow \begin{bmatrix} I & A \\ A^T & I \end{bmatrix} \succeq 0$

$$\begin{bmatrix}1 & 2\end{bmatrix}\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}\begin{bmatrix}5\\ 2\end{bmatrix}\begin{bmatrix}7 & 10\end{bmatrix}\begin{bmatrix}5\\ 11\end{bmatrix}$$

$1\times 2 \quad 2\times 2$

$\|x\|_p$

$\|x\|_1 = \Sigma$

c). A has full column rank

Prove: $\|AB\| \geq \sigma_{min}(A)\,\|B\|$ for any size B

Full column rank $\Rightarrow A^{-1}$ exists

know $\|x^TA\| \geq \sigma_{min}(A) \quad \forall \|x\|=1$ trivially

v_i^T

$(x^TA)^T(x^TA)$

$A^Tx\,x^TA$ same

$(Ax)^T(Ax)$

x^TA^TAx

$$\|AB\| = \max_{\|x\|=1}\|x^TAB\|$$

$$= \max_{\|x\|=1}\|x^TA\|\left\|\frac{x^TA}{\|x^TA\|}B\right\|$$

$$\geq \max_{\|x\|=1}\sigma_{min}(A)\left\|\frac{x^TA}{\|x^TA\|}B\right\|$$

$$= \max_{\|y\|=1}\sigma_{min}(A)\|y^TB\|$$

$$= \sigma_{min}(A)\|B\| \quad \checkmark$$

$$L(\{u_i\}, \{v_j\}) = \sum_{(i,j)\in S} (\langle u_i, v_j \rangle - R_{ij})^2 + \lambda \sum_{i=1}^{n} \|u_i\|_2^2 + \lambda \sum_{j=1}^{m} \|v_j\|_2^2$$

$$\nabla_{u_i} L = \frac{\partial}{\partial u_i} \sum_{(i,j)\in S} (\underbrace{\langle u_i, v_j \rangle}_{u_i^T v_j} - R_{ij})^2 + 2\lambda u_i$$

$$\nabla_{u_i} L = 2 \sum_{j:(i,j)\in S} v_j (v_j^T u_i - R_{ij}) + 2\lambda u_i$$

$$= (UV - R)^2 + \lambda \left(\mathrm{Tr}(U U^T) + \mathrm{Tr}(V^T V) \right)$$

$$= 2(UV - R)V + \lambda \mathrm{Tr}(\nabla_U U^T U)$$

$$= 2(U^T V - R)V + \lambda \mathrm{Tr}(2U)$$

M1 Take R as complete matrix

$$f = \sum_{(i,j)\in S} (\langle u_i, v_j \rangle)^2 - 2\langle u_i, v_j \rangle R_{ij} + R_{ij}^2 + \lambda \sum_{i=1}^{n} \|u_i\|_2^2 + \lambda \sum_{j=1}^{m} \|v_j\|_2^2$$

$$= \sum_{(i,j)\in S} u_i^T v_j u_i^T v_j - 2 u_i^T v_j R_{ij} + R_{ij}^2 + \lambda \sum_{i=1}^{n} u_i^T u_i + \lambda \sum_{j=1}^{m} v_j^T v_j$$

$$\frac{\partial f}{\partial u_i} = \sum_{j:(i,j)\in S} 2 v_j (u_i^T v_j) - 2 v_j R_{ij} + \lambda \sum 2 u_i$$

$$0 = 2\left(\sum_{j:(i,j)\in S} v_j (v_j^T u_i - R_{ij}) + \lambda u_i \right)$$

$$0 = V V^T u_i - \sum_{j:(i,j)\in S} v_j R_{ij} + \lambda u_i$$

$$(V V^T + \lambda I) u_i = \sum_{j:(i,j)\in S} v_j R_{ij}$$

$$u_i = (V V^T + \lambda I)^{-1} \sum_{j:(i,j)\in S} v_j R_{ij}$$

$$= (V V^T + \lambda I) V R_i \qquad V \quad R$$

$$U = (V V^T + \lambda I)^{-1} V R^T$$

By symmetry

$$V = (V V^T + \lambda I) V R$$