ASSIGNMENT NO.1

1. **Importing Libraries**

```
In [1]: %matplotlib inline
        import matplotlib
        import matplotlib.pyplot as plt
        from sklearn.decomposition import PCA
        import numpy as np
        from sklearn.datasets import fetch_mldata
        mnist= fetch_mldata('MNIST original')
```

2. **Storing data into different lists**

```
In [2]: X, y = mnist["data"], mnist["target"]
        X.shape

Out[2]: (70000, 784)
```

3. **Printing random image from training data**

```
In [3]: some_digit= X[36000]
        some_digit_image= some_digit.reshape(28, 28)
        plt.imshow(some_digit_image, cmap= matplotlib.cm.binary,
        interpolation="nearest")
        plt.axis("off")
        plt.show()
```

4. **Separating training and testing data**

```
In [4]: X_train, X_test, y_train, y_test= X[:60000], X[60000:], y[:60000], y[60000:]
```

5. **Shuffling the training and testing data**

```
In [5]: shuffle_index= np.random.permutation(60000)
        X_train, y_train= X_train[shuffle_index], y_train[shuffle_index]
```

6. **Applying Singular Value Decomposition (SVD)**

```
In [6]: X_centered= X -X.mean(axis=0)
        U, s, Vt= np.linalg.svd(X_centered)
        W2 = Vt.T[:, :2]
        X2D = X_centered.dot(W2)
```

7. **Applying Principal Component Analysis for dimension reduction**

```
In [7]: pca= PCA(n_components= 154)
        X_reduced= pca.fit_transform(X_train)
        X_recovered= pca.inverse_transform(X_reduced)
```

8. Observing the variance ratio

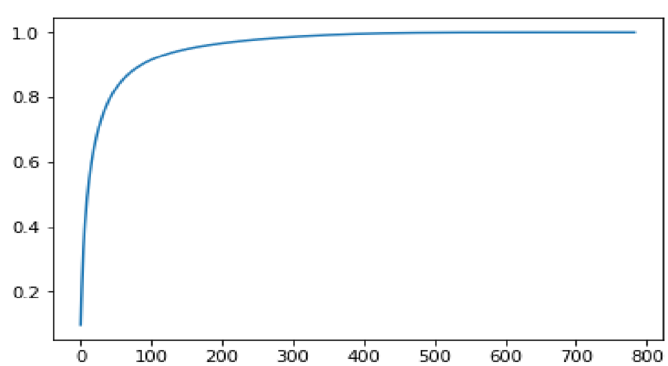
In [8]:
```
pca.explained_variance_ratio_
```

Out[8]:
```
array([ 0.09704664,  0.07095924,  0.06169089,  0.05389419,  0.04868797,
        0.04312231,  0.0327193 ,  0.02883895,  0.02762029,  0.02357001,
        0.0210919 ,  0.02022991,  0.01715818,  0.01692111,  0.01578641,
        0.01482953,  0.01324561,  0.01276897,  0.01187263,  0.01152684,
        0.01066166,  0.01006713,  0.00953573,  0.00912544,  0.00883405,
        0.00839319,  0.00812579,  0.00786366,  0.00744733,  0.00690859,
        0.00658094,  0.00648148,  0.00602615,  0.00586582,  0.00570021,
        0.00543628,  0.00505786,  0.00487859,  0.00481429,  0.00472266,
        0.00456747,  0.00444836,  0.00418501,  0.00398215,  0.00384975,
        0.00375103,  0.00362009,  0.00351591,  0.00340058,  0.00321874,
        0.00319017,  0.00312805,  0.00295983,  0.00288955,  0.0028413 ,
        0.00271436,  0.00269521,  0.00258473,  0.00253771,  0.00244781,
        0.00240506,  0.00239263,  0.00230408,  0.00221532,  0.00213721,
        0.00207225,  0.00203043,  0.00196782,  0.00192852,  0.00188632,
        0.00186977,  0.00181083,  0.00177562,  0.00174898,  0.00165758,
        0.00163894,  0.00161462,  0.00155115,  0.00147612,  0.00143175,
        0.00142093,  0.00141152,  0.00140174,  0.00135734,  0.00133846,
        0.00132395,  0.00130155,  0.00125871,  0.00122827,  0.00121581,
        0.0011703 ,  0.0011487 ,  0.0011324 ,  0.0011088 ,  0.00108998,
        0.00106909,  0.00104189,  0.00103998,  0.00101234,  0.00100512,
        0.00098369,  0.00094954,  0.00094121,  0.00091551,  0.00090752,
        0.0008966 ,  0.00086486,  0.00085464,  0.0008452 ,  0.00082211,
        0.0007907 ,  0.00078517,  0.00078369,  0.00076781,  0.00076357,
        0.00075117,  0.0007346 ,  0.00072591,  0.00071882,  0.00070568,
        0.00069303,  0.00068869,  0.00068239,  0.00067078,  0.00066466,
        0.00064303,  0.0006328 ,  0.00062866,  0.00061916,  0.00059741,
        0.00059633,  0.00058792,  0.0005822 ,  0.00058044,  0.00057543,
        0.00056921,  0.00055845,  0.00054932,  0.00052734,  0.00051308,
        0.00051079,  0.0005035 ,  0.00049426,  0.00049133,  0.00048871,
        0.00047834,  0.00047049,  0.00046386,  0.00046373,  0.00045561,
        0.00044299,  0.00044054,  0.00043244,  0.00043069])
```

9. Plotting the cumulative sum graph the "elbow curve"

In [9]:
```
pca = PCA()
pca.fit(X_train)
cumsum = np.cumsum(pca.explained_variance_ratio_)
d = np.argmax(cumsum>=0.95) + 1
plt.plot(cumsum)
```

Out[9]:
```
[<matplotlib.lines.Line2D at 0x21c1aac1b38>]
```



10. **Random images with all dimensions from training data and recovered data**

In [13]:
```
fig, ax = plt.subplots(2,10, figsize=(20,5), subplot_kw={'xticks':[],'yticks':[]}, gridspec_kw = dict(hspace = 0.1, wspace = 0.1)
for i in range(10):
    ax[0,i].imshow(X_train[20000+i].reshape(28,28), cmap = 'binary_r', interpolation = 'nearest')
    ax[1,i].imshow(X_recovered[20000+i].reshape(28,28), cmap = 'binary_r', interpolation = 'nearest')
ax[0,0].set_ylabel("Full Dimensions")
ax[1,0].set_ylabel("Reconstructed Figures")
```

Out[13]: Text(0,0.5,'Reconstructed Figures')

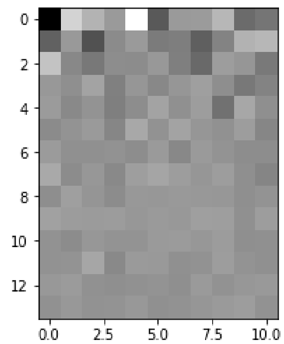
11. **Features of some code elements(components_[i])**

In [15]:
```
fig, axes = plt.subplots(2,10, figsize=(20,5), subplot_kw={'xticks':[],'yticks':[]}, gridspec_kw = dict(hspace = 0.1, wspace = 0.
for i,axe in enumerate(axes.flat):
    axe.imshow(pca.components_[i].reshape(28,28), cmap="bone")
```

12. **Code c for some image x**

```
In [18]: plt.imshow(X_reduced[28930].reshape(14,11), cmap = matplotlib.cm.binary, interpolation = 'nearest')
```

```
Out[18]: <matplotlib.image.AxesImage at 0x2128191b550>
```



13. **Reducing the number of bits of c and reconstructing it. Number of bits are reduced by converting the float value into interger32.**

```
In [83]: Xreduced1 = X_reduced[28000]
nbits1 = Xreduced1.nbytes
Xrecovered1 = pca.inverse_transform(Xreduced1)
Xreduced2 = np.int32(X_reduced[28000])
nbits2 = Xreduced2.nbytes
Xrecovered2 = pca.inverse_transform(Xreduced2)
fig, ax = plt.subplots(2, figsize=(3,5), subplot_kw={'xticks':[],'yticks':[]}, gridspec_kw = dict(hspace = 0.1, wspace = 0.1), sh
ax[0].imshow(Xrecovered1.reshape(28,28), cmap = matplotlib.cm.binary, interpolation = 'nearest')
ax[1].imshow(Xrecovered2.reshape(28,28), cmap = matplotlib.cm.binary, interpolation = 'nearest')
ax[0].set_ylabel("int32 = " + str(nbits2) +" bits")
ax[1].set_ylabel("float = " + str(nbits1) +" bits")
```

```
Out[83]: Text(0,0.5,'float = 1232 bits')
```