DSE 210: Probability and statistics Worksheet 7 — Clustering

For this problem, we'll be using the animals with attributes data set. Go to

http://attributes.kyb.tuebingen.mpg.de (http://attributes.kyb.tuebingen.mpg.de)

and, under "Downloads", choose the "base package" (the very first file in the list). Unzip it and look

over the various text files.

In [1]:
```
%pylab inline
import pandas as pd
import numpy as np
import sklearn as sk
```

```
Populating the interactive namespace from numpy and matplotlib
```

load data

This is a small data set that has information about 50 animals. The animals are listed in classes.txt. For each animal, the information consists of values for 85 features: does the animal have a tail, is it slow, does it have tusks, etc. The details of the features are in predicates.txt. The full data consists of a 50 × 85 matrix of real values, in predicate-matrix-continuous.txt. There is also a binarized version of this data, in predicate-matrix-binary.txt.

In [31]:
```
animals = []
with open('Animals_with_Attributes/classes.txt') as f:
    for line in f.readlines():
        s = line.strip()
        name = s[s.rindex('\t')+1:]
        animals.append(name)
data = np.loadtxt('Animals_with_Attributes/predicate-matrix-continuous.txt')
data.shape
```

Out[31]: (50L, 85L)

Load the real-valued array, and also the animal names, into Python. Run k-means on the data (from sklearn.cluster) and ask for k = 10 clusters. For each cluster, list the animals in it. Does the clustering make sense?

In [93]:
```
from sklearn.cluster import KMeans
estimator = KMeans(init='k-means++', n_clusters=10, n_init=10)
estimator.fit(data)
labels = estimator.labels_
print "labels for the 50 animals\n", labels

category = [[] for i in range(10)]
for i,label in enumerate(estimator.labels_):
    category[label].append(animals[i])
print "animals per cluster:"
category
```

```
labels for the 50 animals
[6 9 3 1 0 0 6 0 3 0 8 1 7 5 7 2 4 3 5 4 2 7 2 3 4 1 1 5 1 1 6 7 0 1 1 3 2
 6 4 6 7 2 7 1 9 0 3 1 2 3]
animals per cluster:
```

Out[93]:
```
[['dalmatian',
  'persian+cat',
  'german+shepherd',
  'siamese+cat',
  'chihuahua',
  'collie'],
 ['beaver',
  'mole',
  'hamster',
  'squirrel',
  'rabbit',
  'bat',
  'rat',
  'weasel',
  'mouse',
  'raccoon'],
 ['moose', 'ox', 'sheep', 'buffalo', 'pig', 'cow'],
 ['killer+whale',
  'blue+whale',
  'humpback+whale',
  'seal',
  'otter',
  'walrus',
  'dolphin'],
 ['spider+monkey', 'gorilla', 'chimpanzee', 'giant+panda'],
 ['hippopotamus', 'elephant', 'rhinoceros'],
 ['antelope', 'horse', 'giraffe', 'zebra', 'deer'],
 ['tiger', 'leopard', 'fox', 'wolf', 'bobcat', 'lion'],
 ['skunk'],
 ['grizzly+bear', 'polar+bear']]
```

Clustering

Now hierarchically cluster this data, using scipy.cluster.hierarchy.linkage. Choose Ward's

method, and plot the resulting tree using the dendrogram method, setting the orientation parameter

to ‘right’ and labeling each leaf with the corresponding animal name.

You will run into a problem: the plot is too cramped because the default figure size is so small. To

make it larger, preface your code with the following:

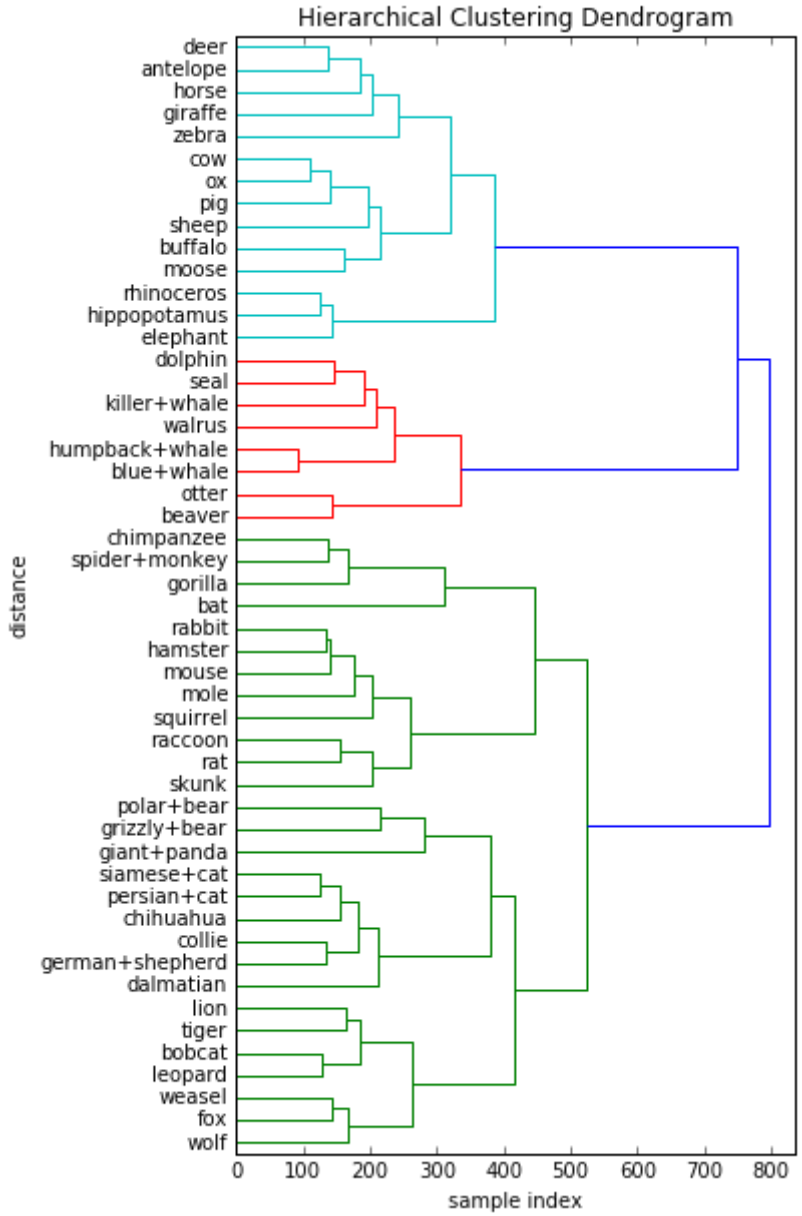
from pylab import rcParams

rcParams[’figure.figsize’] = 5, 10

(or try a different size if this doesn’t seem quite right). Does the hierarchical clustering seem sensible

to you?

In [73]:
```
from scipy.cluster.hierarchy import linkage, dendrogram
z = linkage(data, method='ward', metric='euclidean')

from pylab import rcParams
rcParams['figure.figsize'] = 5, 10

#plt.figure(figsize=(20, 30))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('sample index')
plt.ylabel('distance')
dendrogram(
    z,
    orientation='right',
    leaf_font_size=10.,  # font size for the x axis labels
    labels=animals,
)
plt.show()
```



Worksheet 8 - problem 4, PCA

Recall the animals with attributes data set from Worksheet 7, which has information about 50 animals,

each represented as a vector in R85.

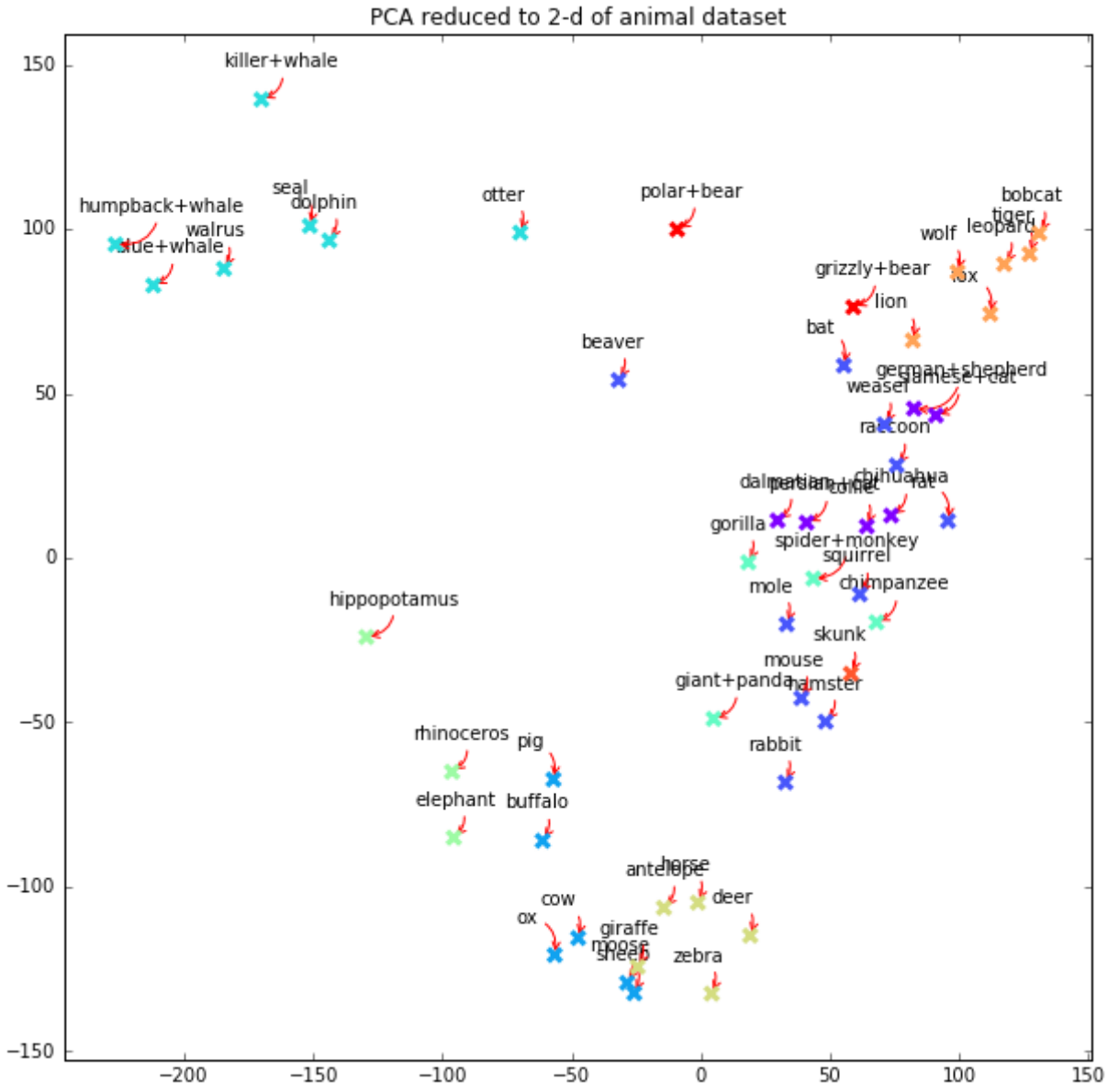
We would like to visualize these animals in 2-d. Show how to do this with a PCA projection from R85 to R2. Show the position of each animal, and label them with their names. (Remember from Worksheet 7 how to enlarge the figure. This time you might want to ask for size 10,10.)

Does this embedding seem sensible to you?

In [67]:
```
from sklearn.decomposition import PCA
reduced_data = PCA(n_components=2).fit_transform(data)
reduced_data.shape
```

Out[67]:
```
(50L, 2L)
```

In [101]:
```
# Step size of the mesh. Decrease to increase the quality of the VQ.
h = .02     # point in the mesh [x_min, x_max]x[y_min, y_max].

# Plot the decision boundary. For that, we will assign a color to each
margin = 20
x_min, x_max = reduced_data[:, 0].min() - margin, reduced_data[:, 0].max() + marg
y_min, y_max = reduced_data[:, 1].min() - margin, reduced_data[:, 1].max() + marg
xx, yy = np.meshgrid(np.arange(x_min, x_max, h), np.arange(y_min, y_max, h))

from pylab import rcParams
rcParams['figure.figsize'] = 10, 10
fig = plt.figure(1)
plt.clf()

colors = matplotlib.cm.rainbow(np.linspace(0, 1, 10))
data_colors = [colors[l] for l in labels]
# The 2-d dataset on 2d map, with color indicating its cluster
plt.title('PCA reduced to 2-d of animal dataset')
plt.scatter(reduced_data[:, 0], reduced_data[:, 1],
            marker='x', s=50, linewidths=3,
            color=data_colors, zorder=10)
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)

# annotate each data point with animal name.
ax = fig.add_subplot(111)
for i in range(reduced_data.shape[0]):
    ax.annotate(animals[i], xy=reduced_data[i],
                xytext=(reduced_data[i][0]-15, reduced_data[i][1]+10),
                #bbox=dict(boxstyle='round,pad=0.2', fc='yellow', alpha=0.3),
                arrowprops=dict(arrowstyle='->', connectionstyle='arc3,rad=-0.5',
plt.show()
```



worksheet8 problem5

a. Compute the fractions of lost information for the MNIST data set, for selecting top eigen values of k = 200, 150, 100, 50, 25.

In [2]:
```
# load MNIST dataset
from sklearn.datasets import fetch_mldata
mnist = fetch_mldata('MNIST original')
print mnist.data.shape
```

```
(70000L, 784L)
```

In [3]:
```
cov_mat = np.cov(mnist.data.T)
print "shape of covariance matrix:", cov_mat.shape
eigenvalues = numpy.linalg.eigvals(cov_mat)
print "number of eigenvalues:", eigenvalues.shape

whole_sum = sum(eigenvalues)
for k in [200, 150, 100, 50, 25]:
    print "lost fraction for k=%d is: %.4f" % (k, sum(eigenvalues[k:])/whole_sum)
```

```
shape of covariance matrix: (784L, 784L)
number of eigenvalues: (784L,)
lost fraction for k=200 is: 0.0335
lost fraction for k=150 is: 0.0515
lost fraction for k=100 is: 0.0850
lost fraction for k=50 is: 0.1746
lost fraction for k=25 is: 0.3070
```

problem5.b

Test whether this is true as follows: for each digit j = 0, 1, 2, . . . , 9,

• Obtain the PCA projection to k dimensions, for k = 200, 150, 100, 50, 25.

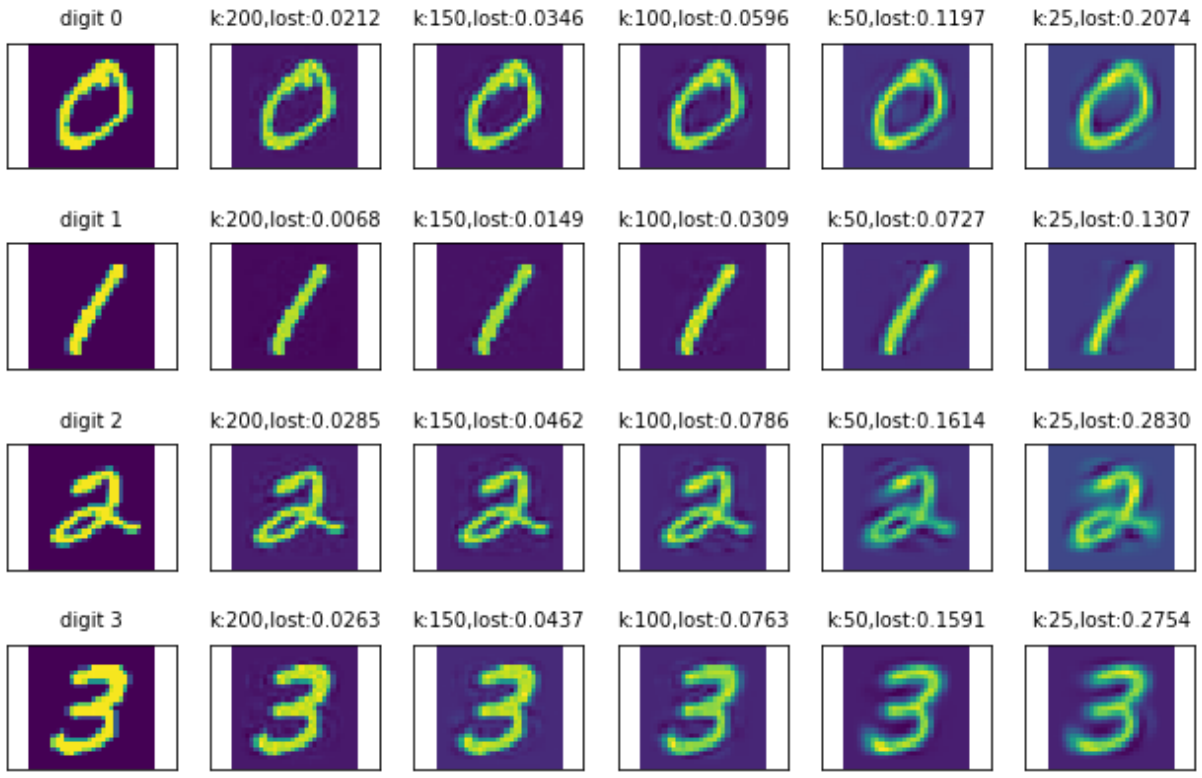
• Compute the fraction Fj (k), for each such value of k.

• Pick a random instance of the digit. Show the original digit as well as its reconstruction at each of these five values of k.

Show all the fractions Fj(k) in a table. Which digit seems to be the most amenable to low-dimensional projection?

In [62]:
```
from sklearn import decomposition
row,col = 10, 6
fig, ax = plt.subplots(row, col, figsize=(col*1.8, row*1.8), sharex='col', sharey
fig.subplots_adjust(hspace=0.6)

def show_digit(ax, data, title):
    ax.axes.get_xaxis().set_visible(False)
    ax.axes.get_yaxis().set_visible(False)
    ax.axes.get_xaxis().set_ticks([])
    ax.axes.get_yaxis().set_ticks([])
    ax.matshow(np.reshape(data,(28,28)))
    ax.set_title(title, size=10)

for j in xrange(10):
    index_j = [i for i in range(len(mnist.target)) if mnist.target[i] == j]
    data_j = mnist.data[index_j]
    cov_mat = np.cov(data_j.T)
    eigenvalues = numpy.linalg.eigvals(cov_mat)

    whole_sum = sum(eigenvalues)

    img_original = data_j[0]  # simply pick 1st image for each digit category
    show_digit(ax[j,0], img_original, "digit %d" %(j))
    for i,k in enumerate([200, 150, 100, 50, 25]):
        lost_fraction = sum(eigenvalues[k:])/whole_sum
        pca = decomposition.PCA(n_components=k)
        pca.fit(data_j)
        mu = np.mean(data_j, axis=0)
        img_after = np.dot(pca.transform(data_j[0].reshape(1,-1)), pca.components
        show_digit(ax[j,i+1], img_after, "k:%d,lost:%.4f" %(k,lost_fraction))
```

```
C:\Users\yanwe\Anaconda2\lib\site-packages\ipykernel\__main__.py:30: ComplexWar
ning: Casting complex values to real discards the imaginary part
```



Final conclusion from above, digit 1 seems to be most amendable to low dimension projection

DSE 210: Probability and statistics **Winter 2015**

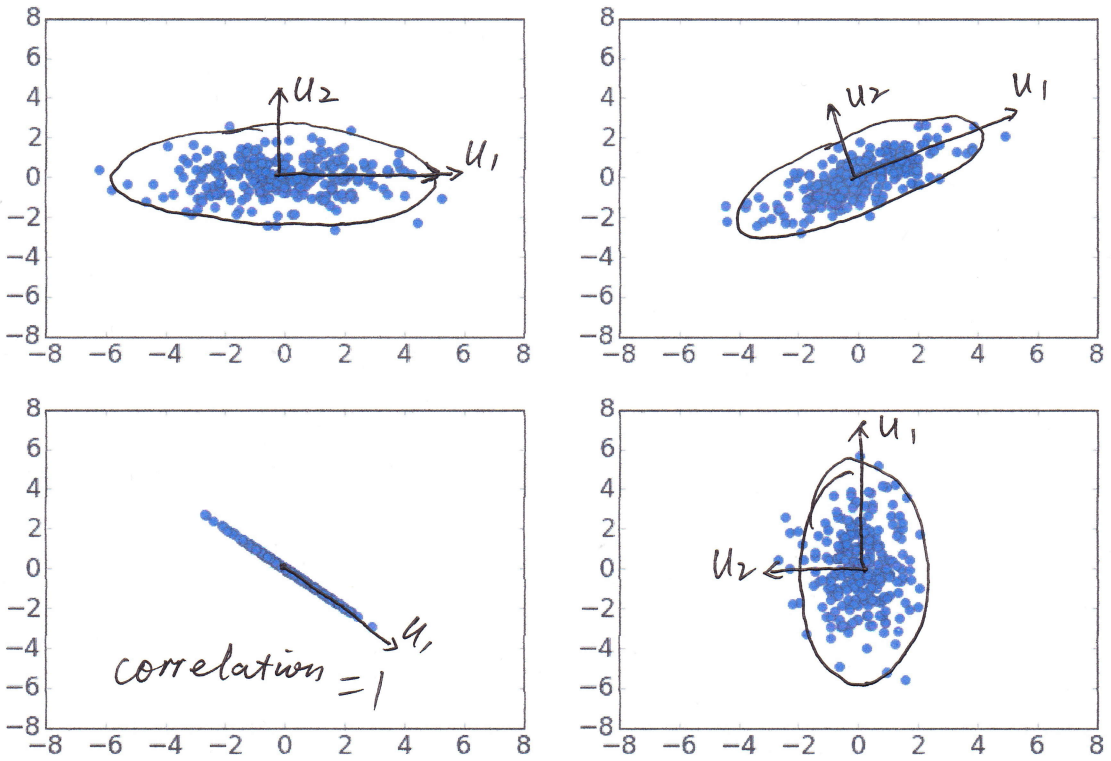
Worksheet 8 — Matrix factorization

1. Is the following set of vectors an orthonormal basis of $\mathbb{R}^3$? Explain why or why not.

$$\begin{pmatrix}3\\4\\0\end{pmatrix}, \begin{pmatrix}4\\-3\\0\end{pmatrix}, \begin{pmatrix}0\\0\\1\end{pmatrix}$$

No. They're orthogonal but not all normalized.

2. The following four figures show different 2-dimensional data sets. In each case, make a rough sketch of an ellipsoidal contour of the covariance matrix and indicate the directions of the first and second eigenvectors (mark which is which).



3. Let $u_1, u_2 \in \mathbb{R}^p$ be two vectors with $\|u_1\| = \|u_2\| = 1$ and $u_1 \cdot u_2 = 0$. Define

$$U = \begin{pmatrix} \uparrow & \uparrow \\ u_1 & u_2 \\ \downarrow & \downarrow \end{pmatrix}$$

(a) What are the dimensions of each of the following?

- U $P \times 2$
- U^T $2 \times P$
- UU^T $P \times P$
- $u_1 u_1^T$ $P \times P$

(b) What are the differences, if any, between the following four projections?

① ② ③ ④

- $x \mapsto (u_1 \cdot x, u_2 \cdot x)$
- $x \mapsto (u_1 \cdot x)u_1 + (u_2 \cdot x)u_2$
- $x \mapsto U^T x$
- $x \mapsto UU^T x$

① ③ are same, they transfer x to new coordinates under U

② ④ are same, they project back to standard basis

4. Recall the *animals with attributes* data set from Worksheet 7, which has information about 50 animals, each represented as a vector in $\mathbb{R}^{85}$.

   We would like to visualize these animals in 2-d. Show how to do this with a PCA projection from $\mathbb{R}^{85}$ to $\mathbb{R}^2$. Show the position of each animal, and label them with their names. (Remember from Worksheet 7 how to enlarge the figure. This time you might want to ask for size 10,10.)

   Does this *embedding* seem sensible to you? See python notebook

5. In lecture, we looked at the effect of projecting the MNIST data set of handwritten digits to lower-dimension: from 784 to 200, 150, 100, 50 dimensions. We found that the reconstruction error was fairly low for 150-dimensional projections, but noticeable for 50-dimensional projections.

   We now investigate these issues further. See python notebook.

   (a) Let $X \in \mathbb{R}^p$ have covariance matrix Σ. Suppose the eigenvalues of Σ are $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_p$, and suppose the corresponding eigenvectors are $u_1, u_2, \ldots, u_p$. Then it can be shown that X has an overall variance of $\lambda_1 + \cdots + \lambda_p$, and that when X is projected onto the top k eigenvectors, the residual variance (the information that gets lost) is $\lambda_{k+1} + \cdots + \lambda_p$. Therefore, for this projection, the fraction of lost information, intuitively speaking, is

   $$F(k) = \frac{\lambda_{k+1} + \cdots + \lambda_p}{\lambda_1 + \cdots + \lambda_p}$$

   Compute these fractions for the MNIST data set, for $k = 200, 150, 100, 50, 25$.

   (b) Suppose we are allowed a different projection for each digit. We would then expect that we can project to an even lower dimension while maintaining roughly the same amount of information. Test whether this is true as follows: for each digit $j = 0, 1, 2, \ldots, 9$,

   - Obtain the PCA projection to k dimensions, for $k = 200, 150, 100, 50, 25$.
   - Compute the fraction $F_j(k)$, for each such value of k.
   - Pick a random instance of the digit. Show the original digit as well as its reconstruction at each of these five values of k. (Note: the original images have pixel values in the range 0-255, but this might not be true of the reconstructions; therefore, you may need to clip the reconstructed pixels to lie within this range before displaying the image.)

   Show all the fractions $F_j(k)$ in a table. Which digit seems to be the most amenable to low-dimensional projection?