Foundations of Data Mining: Assignment 3

Please complete all assignments in this notebook. You should submit this notebook, as well as a PDF version (See File > Download as).

Deadline: Thursday, March 29, 2018

In [2]:
```
from IPython.display import HTML
HTML('''<style>html, body{overflow-y: visible !important} .CodeMirror{min-width:105% !important;} .rise-ena
%matplotlib inline



import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import openml as oml
import mglearn
import os
from cycler import cycler
from pprint import pprint

# Silence warnings
import warnings
warnings.simplefilter(action="ignore", category=FutureWarning)
warnings.simplefilter(action="ignore", category=UserWarning)
warnings.simplefilter(action="ignore", category=RuntimeWarning)
from sklearn.model_selection import *
from sklearn import svm

%matplotlib inline
from preamble import *

import sklearn.decomposition as deco
import sklearn.manifold as manifold

plt.rcParams['savefig.dpi'] = 100 # This controls the size of your figures
# Comment out and restart notebook if you only want the last output of each cell.
InteractiveShell.ast_node_interactivity = "all"
```

PCA and Isomap (5 Points, 1+2+2)

Apply PCA and Isomap to images of handwritten digits (see below). You may use sklearn.decomposition and sklearn.manifold.

a)

Compute the first two components of the data using PCA. Make a scatter plot of the data in the first two components of PCA indicating class with color.

b)

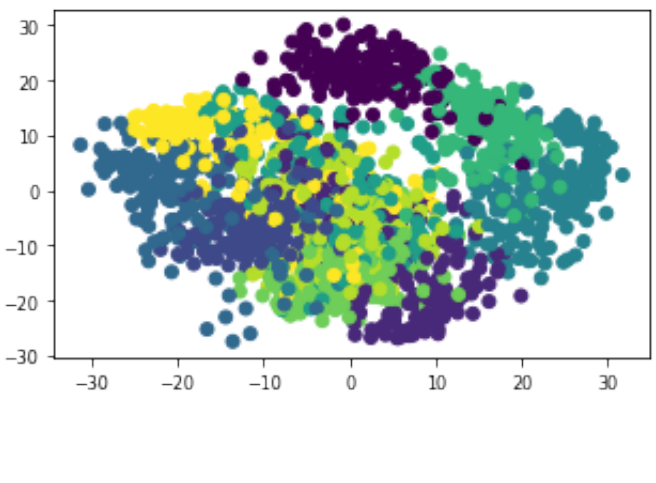
Compute an Isomap embedding with two components with nr_neighbors={5, 50, N-1} (three separate embeddings). For each of the Isomap embeddings, apply the function "align" (see below) with "ref_data" as your computed pca embedding and "data" as the isomap embedding. Show a scatter plot of each of the aligned isomap embeddings.

c)

Visually compare how well the classes are separated in the different scatter plots. What is the effect of changing the number of neighbors on the score computed in the alignment function? What does it mean if the score is zero? When do you expect the score to become zero and why?

In [3]:
```
# Load the data set
from sklearn import datasets
digits = datasets.load_digits(n_class=10)
X = digits.data
y = digits.target
N=len(X)

# Align a data set with a reference data set minimizing l_1 error
# Returns aligned data set and alignment error
def align(ref_data, data):

    transformations = np.asarray([
        [[0,1],[1,0]],
        [[0,-1],[1,0]],
        [[0,1],[-1,0]],
        [[0,-1],[-1,0]],
        [[1,0],[0,1]],
        [[1,0],[0,-1]],
        [[-1,0],[0,1]],
        [[-1,0],[0,-1]]
    ])

    score = []
    for i in range(0,8):
        transf_data =   np.matmul(data, transformations[i])
        score.append(np.linalg.norm( transf_data - ref_data, ord=1) )

    idx = np.argmin(score)
    transf_data = np.matmul(data,transformations[idx])

    print("Aligned the data sets. Score is {0:10.1f}  ".format(score[idx]))

    return transf_data, score[idx];
```

In [4]:
```
from sklearn.decomposition import PCA
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn import datasets

pca = PCA(n_components=2)
pca_digits = pca.fit_transform(X, y=None)

plt.scatter(pca_digits[:,0], pca_digits[:,1], c=y);
```



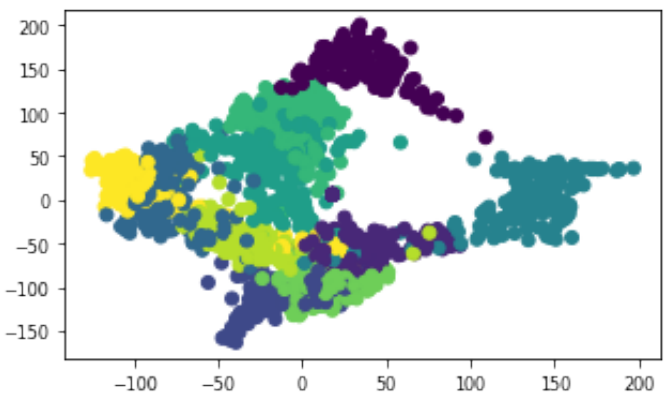
In [5]:
```
from sklearn.manifold import Isomap

isomap_5 = Isomap(n_neighbors = 5, n_components = 2)
isomap_50 = Isomap(n_neighbors = 50, n_components = 2)
isomap_N = Isomap(n_neighbors = N-1, n_components = 2)

digits_isomap_5 = isomap_5.fit_transform(X)
aligned_5 = align(pca_digits, digits_isomap_5)
arr_5, sc_5 = aligned_5

digits_isomap_50 = isomap_50.fit_transform(X)
aligned_50 = align(pca_digits, digits_isomap_50)
arr_50, sc_50 = aligned_50

digits_isomap_N = isomap_N.fit_transform(X)
aligned_N = align(pca_digits, digits_isomap_N)
arr_N, sc_N = aligned_N;
```

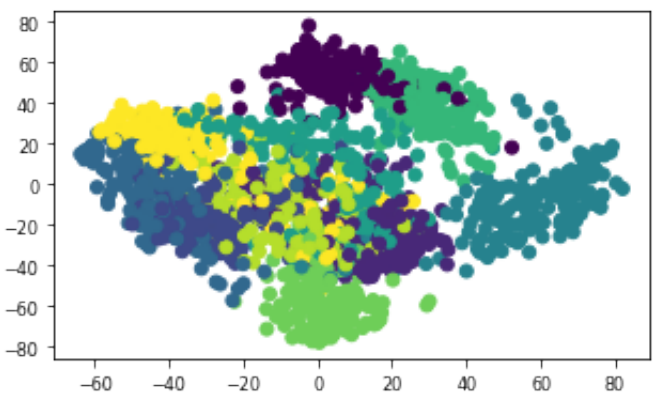
```
Aligned the data sets. Score is   102234.2
Aligned the data sets. Score is    31067.5
Aligned the data sets. Score is        0.0
```

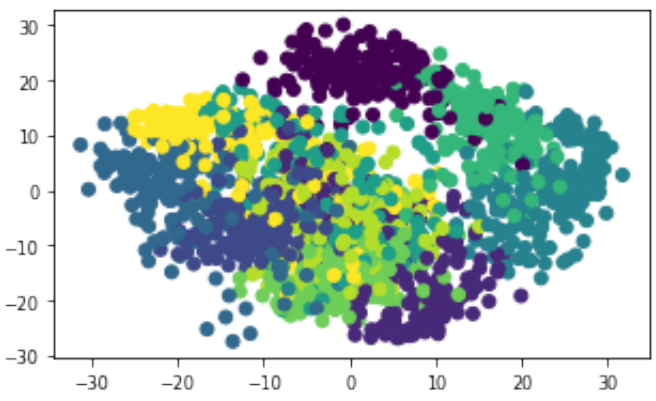
In [6]:
```
print(arr_5)
print(pca_digits)
```

```
[[  22.185  167.272]
 [  51.308  -48.715]
 [  25.821 -100.72 ]
 ...
 [ -23.294  -51.436]
 [ -73.328   -0.221]
 [ -36.237  -11.366]]
[[ -1.259  21.275]
 [  7.958 -20.769]
 [  6.992  -9.956]
 ...
 [ 10.801  -6.96 ]
 [ -4.872  12.424]
 [ -0.344   6.366]]
```

In [7]:
```
plt.scatter(arr_5[:,0], arr_5[:,1], c=y)
plt.show()
plt.scatter(arr_50[:,0], arr_50[:,1], c=y)
plt.show()
plt.scatter(arr_N[:,0], arr_N[:,1], c=y)
plt.show();
```







Answer to 1c.

Increasing the number of neighbors causes the different classes to overlap more. It also decreases the score computed in the alignment function, eventually making it go to 0 when the number of neighbors reaches the total number of datapoints. This means that the more neighbors, the more similar the isomap result is to PCA. A score of 0 for N-1 neighbors thus means that it gives an identical result to PCA, as the score computes the distance between each isomap and corresponding pca point. This can also be seen from the plot, as the plot generated from N-1 neighbors looks identical to the scatterplot of the data after PCA. We can conclude that Isomap for N-1 neighbors is identical to a PCA method.

Classical Multidimensional Scaling (6 Points, 1+2+2+1)

Show that for mean-centered data sets we can recover inner products using pairwise distance information only. This is used by the isomap embedding algorithm.

We are given all squared pairwise distances of an otherwise unknown point set $\mathbf{p}_1, \ldots, \mathbf{p}_n \in \mathbb{R}^d$, i.e., we are given for all $1 \leq i, j \leq n$ the values

$$d_{ij} = \|\mathbf{p}_i - \mathbf{p}_j\|^2.$$

We assume that the point set is mean-centered, that is

$$\sum_{i=1}^{n} \mathbf{p}_i = \vec{\mathbf{0}}.$$

(where $\vec{\mathbf{0}}$ is the vector of zeros)

In the following, $\langle \mathbf{p}_i, \mathbf{p}_j \rangle$ denotes the inner product of $\mathbf{p}_i$ and $\mathbf{p}_j$. Prove that the following holds true for mean-centered point sets:

$$-2\langle \mathbf{p}_i, \mathbf{p}_j \rangle = d_{ij} - \sum_{\ell=1}^{n} \frac{d_{i\ell}}{n} - \sum_{\ell=1}^{n} \frac{d_{j\ell}}{n} + \sum_{k=1}^{n} \sum_{\ell=1}^{n} \frac{d_{k\ell}}{n^2}$$

You may use the following steps in your derivation.

a)

Expand d_{ij} to yield an expression of $\langle \mathbf{p}_i, \mathbf{p}_j \rangle$, $\|\mathbf{p}_i\|^2$ and $\|\mathbf{p}_j\|^2$.

b)

Show that the following holds for any $\mathbf{q} \in \mathbb{R}^d$:

$$\sum_{1 \leq i \leq n} \langle \mathbf{p}_i, \mathbf{q} \rangle = 0$$

c)

Prove that

$$\|\mathbf{p}_i\|^2 = \sum_{\ell=1}^{n} \frac{d_{i\ell}}{n} - \sum_{k=1}^{n} \sum_{\ell=1}^{n} \frac{d_{k\ell}}{2n^2}$$

d)

Combine the steps in your proof.

Classical Multidimensional Scaling Proof

We have the following expression for d_{ij}:

$$d_{ij} = \|\mathbf{p}_i - \mathbf{p}_j\|^2.$$

If we extend d_{ij}, we get the following:

$$d_{ij} = (\mathbf{p}_{i1} - \mathbf{p}_{j2})^2 + (\mathbf{p}_{i2} - \mathbf{p}_{j2})^2 + \ldots + (\mathbf{p}_{in} - \mathbf{p}_{in})^2 = \mathbf{p}_{i1}^2 + \mathbf{p}_{j1}^2 - 2\mathbf{p}_{i1}\mathbf{p}_{j1} + \ldots + \mathbf{p}_{in}^2 + \mathbf{p}_{jn}^2 - 2\mathbf{p}_{in}\mathbf{p}_{jn} = \|\mathbf{p}_i\|^2 + \|\mathbf{p}_j\|^2 - 2\langle \mathbf{p}_i, \mathbf{p}_j \rangle$$

Therefore, we can translate into the following expression:

$$-2\langle \mathbf{p}_i, \mathbf{p}_j \rangle = d_{ij} - \|\mathbf{p}_i\|^2 - \|\mathbf{p}_j\|^2$$

It is assumed that the point set is mean-centered, that is

$$\sum_{i=1}^{n} \mathbf{p}_i = \vec{\mathbf{0}}.$$

For any $\mathbf{q} \in \mathbb{R}^d$, we can therefore say that the following holds:

$$\sum_{1 \le i \le n} \langle \mathbf{p}_i, \mathbf{q} \rangle = \sum_{1 \le i \le n} \sum_{1 \le k \le n} \mathbf{p}_{ik}\mathbf{q}_k = \sum_{1 \le i \le n} \mathbf{p}_{ik} \sum_{1 \le k \le n} \mathbf{q}_k = 0$$

Because of the mean-centeredness, the mean center of $\mathbf{p}_{il}$ can be expressed by

$$\sum_{\ell=1}^{n} \frac{d_{i\ell}}{n}$$

Similarly, the mean center of $\mathbf{p}_{jl}$ can be expressed by

$$\sum_{\ell=1}^{n} \frac{d_{j\ell}}{n}$$

However, to get to an expression for $\|\mathbf{p}_i\|^2$, we still need to look at the mean center of $\mathbf{p}_{kl}$. This can be expressed by

$$\sum_{k=1}^{n} \sum_{\ell=1}^{n} \frac{d_{k\ell}}{n^2}$$

To get to $\|\mathbf{p}_i\|^2$, half of this needs to be subtracted from the mean center, yielding the following expression:

$$\|\mathbf{p}_i\|^2 = \sum_{\ell=1}^{n} \frac{d_{i\ell}}{n} - \sum_{k=1}^{n} \sum_{\ell=1}^{n} \frac{d_{k\ell}}{2n^2}$$

Similarly,

$$\|\mathbf{p}_j\|^2 = \sum_{\ell=1}^{n} \frac{d_{j\ell}}{n} - \sum_{k=1}^{n} \sum_{\ell=1}^{n} \frac{d_{k\ell}}{2n^2}$$

When combining the above steps, we get the following expression:

$$-2\langle \mathbf{p}_i, \mathbf{p}_j \rangle = d_{ij} - \|\mathbf{p}_i\|^2 - \|\mathbf{p}_j\|^2 = d_{ij} - (\sum_{\ell=1}^{n} \frac{d_{i\ell}}{n} - \sum_{k=1}^{n} \sum_{\ell=1}^{n} \frac{d_{k\ell}}{2n^2}) - (\sum_{\ell=1}^{n} \frac{d_{j\ell}}{n} - \sum_{k=1}^{n} \sum_{\ell=1}^{n} \frac{d_{k\ell}}{2n^2})$$
$$= d_{ij} - \sum_{\ell=1}^{n} \frac{d_{i\ell}}{n} - \sum_{\ell=1}^{n} \frac{d_{j\ell}}{n} + \sum_{k=1}^{n} \sum_{\ell=1}^{n} \frac{d_{k\ell}}{2n^2} + \sum_{k=1}^{n} \sum_{\ell=1}^{n} \frac{d_{k\ell}}{2n^2} = d_{ij} - \sum_{\ell=1}^{n} \frac{d_{i\ell}}{n} - \sum_{\ell=1}^{n} \frac{d_{j\ell}}{n} + \sum_{k=1}^{n} \sum_{\ell=1}^{n} \frac{d_{k\ell}}{n^2}$$

Q.E.D.

Locality-sensitive hashing (4 Points, 2+1+1)

a)

Prove that if the Jaccard Similarity of two sets is 0, then minhashing always gives a correct estimate of the Jaccard similarity

for $sim_{jac}(S_i, S_j) = 0$

Define for any representations of S_i and S_j where

x is number of rows where S_i and S_j both have 1

y is number of rows where either S_i or S_j have 1 and the other has 0

z is number of rows where both S_i and S_j have 0

as $z >> x, y$ these rows are ignored in calculation of sim_{jac}

minhashing gives $E[sim_{jac}(S_i, S_j)]$ by performing multiple random permutations m_i from $\{m_1, m_2, ..., m_n\}$

this permutation returns $X_\ell = 1 \; for \; (m_\ell(S_i)=m_\ell(S_j)) \; , 0 \; elsewhere$

this permutation is performed k times and $E[sim_{jac}(S_i, S_j)] = \sum_{\ell=1}^{k} \frac{X_\ell}{k}$

$P[X_\ell = 1] = \frac{x}{x+y}$ for $sim_{jac}(S_i, S_j) = 0$ there are no rows where S_i and S_j both have $'1'$

this gives x = 0 and therefore $P[X_\ell = 1] = 0$

for all values of $\ell \; X_\ell = 0$

therefore for any k number of permutations m_ℓ

$E[sim_{jac}(S_i, S_j)] = \sum_{\ell=1}^{k} \frac{X_\ell}{k} = 0$

b)

Let H be a family of (d_1, d_2, p_1, p_2)-locality-sensitive hash functions. Assume that $p_2 = 0$ and assume we have a total numer of m hash functions from this family available. Which combination of AND-constructions and OR-constructions should we use to maximally amplify the hash family?

for H being a family of (d_1, d_2, p_1, p_2)-locality-sensitive hash functions with m number of hash functions and $p_2 = 0$.

an n-way AND-operation would result in family H' of (d_1, d_2, p_1^n, p_2^n)-locality-sensitive hash functions.

an r-way OR-operation would result in family H'' of $(d_1, d_2, 1-(1-p_1)^r, 1-(1-p_2)^r)$-locality-sensitive hash functions, as the probability that a hash function would nominate a pair as a candidate (x, y) with $\|x - y\| = d_1$ is p_1 , that it would NOT nominate is $1 - p_1$, that r number of functions would NOT nominate is $(1 - p_1)^r$, and therefore that any number r above 1 would nominate the pair is $1 - (1 - p_1)^r$.

as $p_2 = 0$, amplification of the family would simply entail raising p_1 to its maximum value possible within family H.

as $0 \leq p_1 \leq 1$ and $1 \leq n \leq m \; p_1^n < p_1$ for all values $n > 1$

as $1 \leq r \leq m \; 1 - (1 - p_1)^r > p_1$ for all values $r > 1$ with the probability rising for increasing r

therefore to maximally amplifying the hash family a 1-way AND-operation followed by an m-way OR-operation should be performed.

(c)

Let H be a family of (d_1, d_2, p_1, p_2)-locality-sensitive hash functions. Assume that $p_2 = \frac{1}{n}$ and assume we have n data points $\mathbf{P}$ which are stored in a hash table using a randomly chosen function h from H. Given a query point $\mathbf{q}$, we retrieve the points in the hash bucket with index $h(\mathbf{q})$ to search for a point which has small distance to $\mathbf{q}$. Let X be a random variable that is equal to the size of the set

$$\{\mathbf{p} \in \mathbf{P} \; : \; h(\mathbf{p}) = h(\mathbf{q}) \; \wedge \; d(\mathbf{p}, \mathbf{q}) \geq d_2 \}$$

which consists of the false positives of this query. Derive the expected number of false-positives $E[X]$.

$E[X] = E[\sum_{i=0}^{n} Y_i] = \sum_{i=0}^{n} E[Y_i]$

$E[Y_i] = Y_\mu$

$\sum_{i=0}^{n} E[Y_i] = n * Y_\mu$

$Y_\mu = P[(h(p) = h(q)) \wedge (d(\mathbf{p}, \mathbf{q}) \geq d_2)] = P[(h(p) = h(q))] * P[d(\mathbf{p}, \mathbf{q}) \geq d_2]$

for n datapoints and $m_{p,q}$ is number of datapoints for which $d(p, q) < d_2$ and $r_{p,q}$ = 1-$m_{p,q}$

$Y_\mu = P[(h(p) = h(q))] * P[d(\mathbf{p}, \mathbf{q}) \geq d_2]$

$P[d(\mathbf{p}, \mathbf{q}) \geq d_2] = \frac{r_{p,q}}{n}$

$P[(h(p) = h(q))] \leq p_2$

$p_2 = \frac{1}{n}$

$Y_\mu \leq \frac{1}{n} * \frac{r_{p,q}}{n}$

$Y_\mu \leq \frac{r_{p,q}}{n^2}$

$E[X] \leq n * \frac{r_{p,q}}{n^2}$

$$E[X] \leq \frac{r_{p,q}}{n}$$

In [ ]: