isomap

December 14, 2020

Appendix project section: This is not a part of the main-project, but while reading up about different ML algorithms, I stubmled upon ISOMAP and given how ineteresting it looked, I decided to try and implement it. Given that I performed this on the MNSIT dataset, I think putting this in my project's appendix section would be a useful insight, although not contributing to my project

[4]:
```
%matplotlib inline
import gzip
import struct
import matplotlib.pyplot as plt
import numpy as np
from sklearn import manifold


# This method was created by Tyler Neylon to read MNIST's idx file format into␣
↪numpy arrays.
# Source: https://gist.github.com/tylerneylon/ce60e8a06e7506ac45788443f7269e40

def read_idx(filename):
    with gzip.open(filename) as f:
        zero, data_type, dims = struct.unpack('>HBB', f.read(4))
        shape = tuple(struct.unpack('>I', f.read(4))[0] for d in range(dims))
        return np.fromstring(f.read(), dtype=np.uint8).reshape(shape)
```

[5]:
```
trainingdata_raw = read_idx("train-images-idx3-ubyte.gz")
traininglabels_raw = read_idx("train-labels-idx1-ubyte.gz")
```

```
/opt/conda/lib/python3.7/site-packages/ipykernel_launcher.py:16:
DeprecationWarning: The binary mode of fromstring is deprecated, as it behaves
surprisingly on unicode inputs. Use frombuffer instead
  app.launch_new_instance()
```

[6]:
```
print("Size of the raw data aray is: ", trainingdata_raw.shape)
```

```
Size of the raw data aray is:  (60000, 28, 28)
```

[19]:
```
# Converting into one flat vector
vector_training_data = np.reshape(trainingdata_raw, (60000, 784))
```

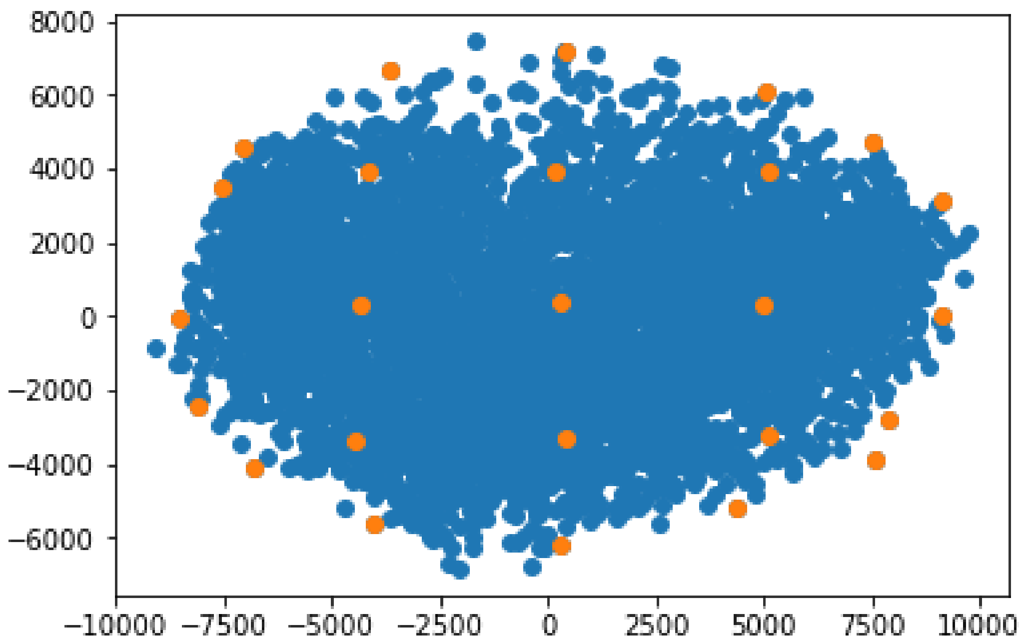
```
<class 'numpy.ndarray'>
```

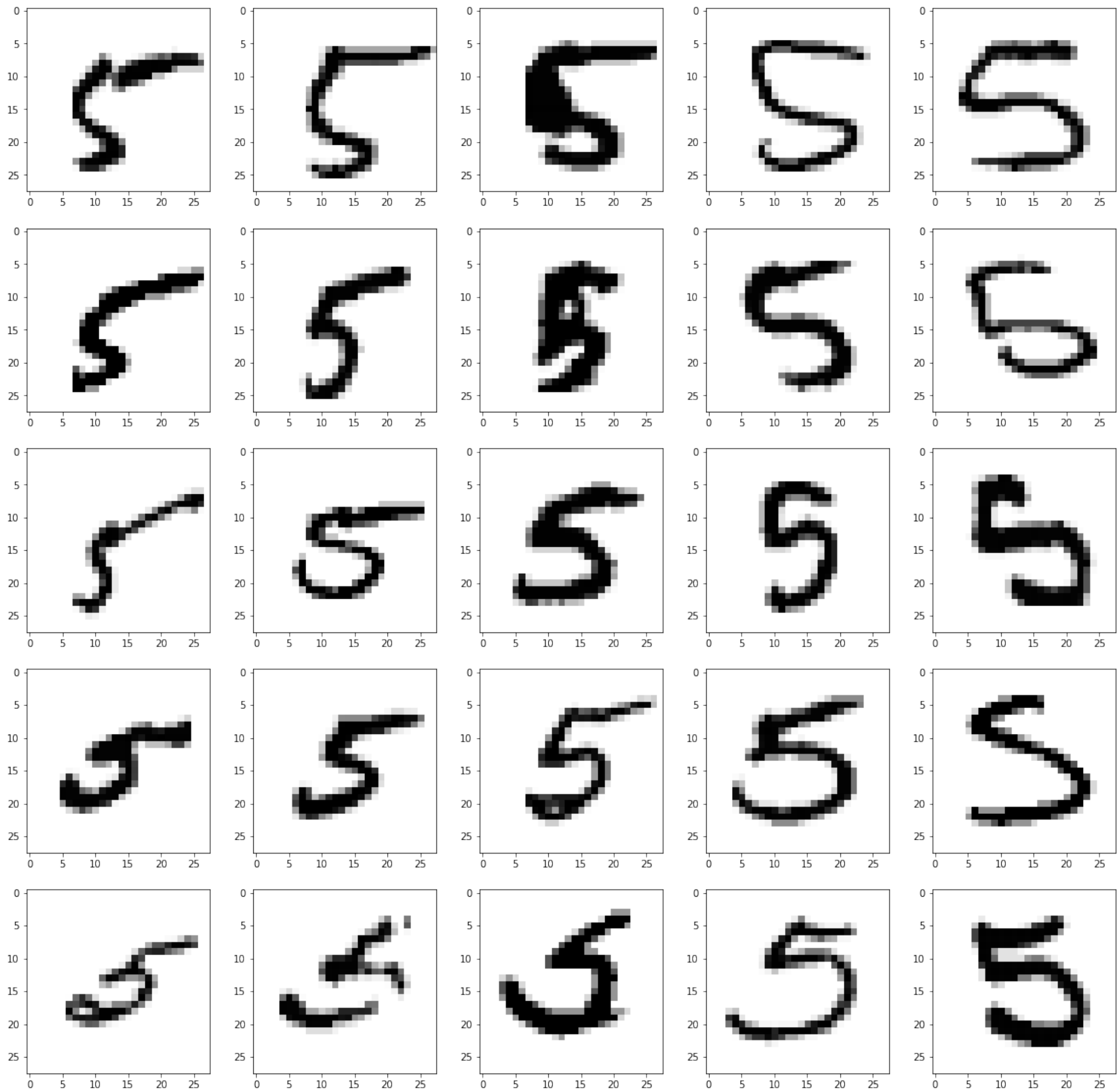
[20]:
```
# Observing Corelations and Trends using IsoMap one digit at a time. This makes␣
↪the 60k data size into a more
# digestible size and will allow us to look at trends more easily
# The two main parameters to define are n_neighbours and n_components
# n_neighbours :  is the number of nearest neighbours we want to consider for␣
↪dimensionality reduction
# n_components : is the number of dimensions we want to extract

#We only want to observe the digits correspondin to the number 5
A_matrix = vector_training_data[traininglabels_raw == 5]

#Performing isomap woth 5 n_neighbours and 2 n_components, so the data reduced␣
↪to 2 dimensions
A_dim_reduction = manifold.Isomap(n_neighbors = 5, n_components=2).
↪fit_transform(A_matrix)
```

[17]:
```
# In order to correlate this plot with some features of the digit 5, I created␣
↪a grid
# to select landmark points, for example, here I selected 5 approximately
#evenly spaced points in each dimension for a total of 25 points. Later, I will␣
↪plot the digits associated with them
# in order to observe any visual trends that the isomap technoque learned

def points(A_dim_reduction, n, m):
    #Creating a grid of evenly spaced points along the x and y axis
    gridx, gridy = np.meshgrid(np.linspace(np.min(A_dim_reduction[:, 0]), np.
↪max(A_dim_reduction[:, 0]), n), np.linspace(np.min(A_dim_reduction[:, 1]),␣
↪np.max(A_dim_reduction[:, 1]), m))
    #Defining a list to be populated by the indices of the selected points of␣
↪size nxm
    idx = [0]*(n*m)
    for i, x, y in zip(range(n*m), gridx.flatten(), gridy.flatten()):
        idx[i] = int(np.sum(np.abs(A_dim_reduction-np.array([x,y]))**2,␣
↪axis=-1).argmin())
    return idx
points_of_concern = points(A_dim_reduction, 5, 5)
```

[18]:
```
#Graphing all the 2 dimensional points selected
plt.scatter(A_dim_reduction[:, 0], A_dim_reduction[:,1])
#Choosing 25 landmark points to plot
plt.scatter(A_dim_reduction[points_of_concern, 0],␣
↪A_dim_reduction[points_of_concern,1])

#Plotting images associated with each of the landmark points
fig = plt.figure(figsize = (20,20))
for i in range(len(points_of_concern)):
```

```
    ax = fig.add_subplot(5, 5, i+1)
    #Reshaping np arrays back into 2D arrays
    imgplot = ax.imshow(np.reshape(A_matrix[points_of_concern[i]], (28, 28)),␣
 ↪cmap = plt.cm.get_cmap("Greys"))
    #Smooths out image
    imgplot.set_interpolation("nearest")
plt.show()
```





0.1 Above is the data derived from the landmark points obtained from the grid. By simply looking at the points from left to right, we can observe that the tilt angle of the digits goes from being tilted towards the right, to being centred, to being tilted towards the left. The Isomap seems to have learned along the first dimension, the tilt associated with the digit being written. If we observe the data on the top vs at the bottom, it is clear that the digits on the top have a greater asymmetry in the circles of the number, and the digits at the lower end are more proportional.