Visualizing MNIST using t-SNE

October 25, 2017

1 Visualizing MNIST using t-SNE

The objective of this Notebook is to visualize the MNIST data set in two and three dimensions, using the projection onto this lower dimensional spaces given by the technique t-SNE.

```
In [30]: #Imports and a little bit of setup
         from __future__ import print_function
         import numpy as np
         import matplotlib.pyplot as plt
         from mpl_toolkits.mplot3d import Axes3D
         from mnist import MNIST
         import math

         # import jtplot submodule from jupyterthemes
         from jupyterthemes import jtplot

         # currently installed theme will be used to
         # set plot style if no arguments provided
         jtplot.style(theme = 'default')
         #jtplot.style()

         #A little bit of matplotlib magic so that the images can be showed on the IPython not
         %matplotlib inline
         plt.rcParams['figure.figsize'] = (10.0, 8.0) # set default size of plots
         plt.rcParams['image.interpolation'] = 'nearest'
         plt.rcParams['image.cmap'] = 'gray'

         #Small function that helps displays images
         def imshow_noax(img, normalize=True):
             """ Tiny helper to show images as uint8 and remove axis labels """
             if normalize:
                 img_max, img_min = np.max(img), np.min(img)
                 img = 255.0 * (img - img_min) / (img_max - img_min)
             plt.imshow(img.astype('uint8'))
             plt.gca().axis('off')
```

```
In [6]: #Imports the data and converts it to numpy arrays
        mndata = MNIST('../python-mnist/data')
```

```
X, labels = mndata.load_testing()

#converts to numpy array
X = np.array(X)
labels = np.array(labels)

subsample_idices = np.random.choice(X.shape[0], 2000)
X = X[subsample_idices,:]
labels = labels[subsample_idices]

#Converts to binary the selcted sample
X = np.around(1-(X/255))

N, D = X.shape

#Declares the cmap for coloring
#COlors
selected_colors = ['red','greenyellow','blue','pink','yellow','orange','lightcyan','da
cmap = plt.cm.jet
# create the new map
cmap = cmap.from_list('Custom cmap', selected_colors, 10)
```

2 t-SNE

This authors of this technique offer a free python implementation on their website: *https://lvdmaaten.github.io/tsne/*. The script was downloaded and is imported into this notebook (some small syntax changes where made for it to run on python3)

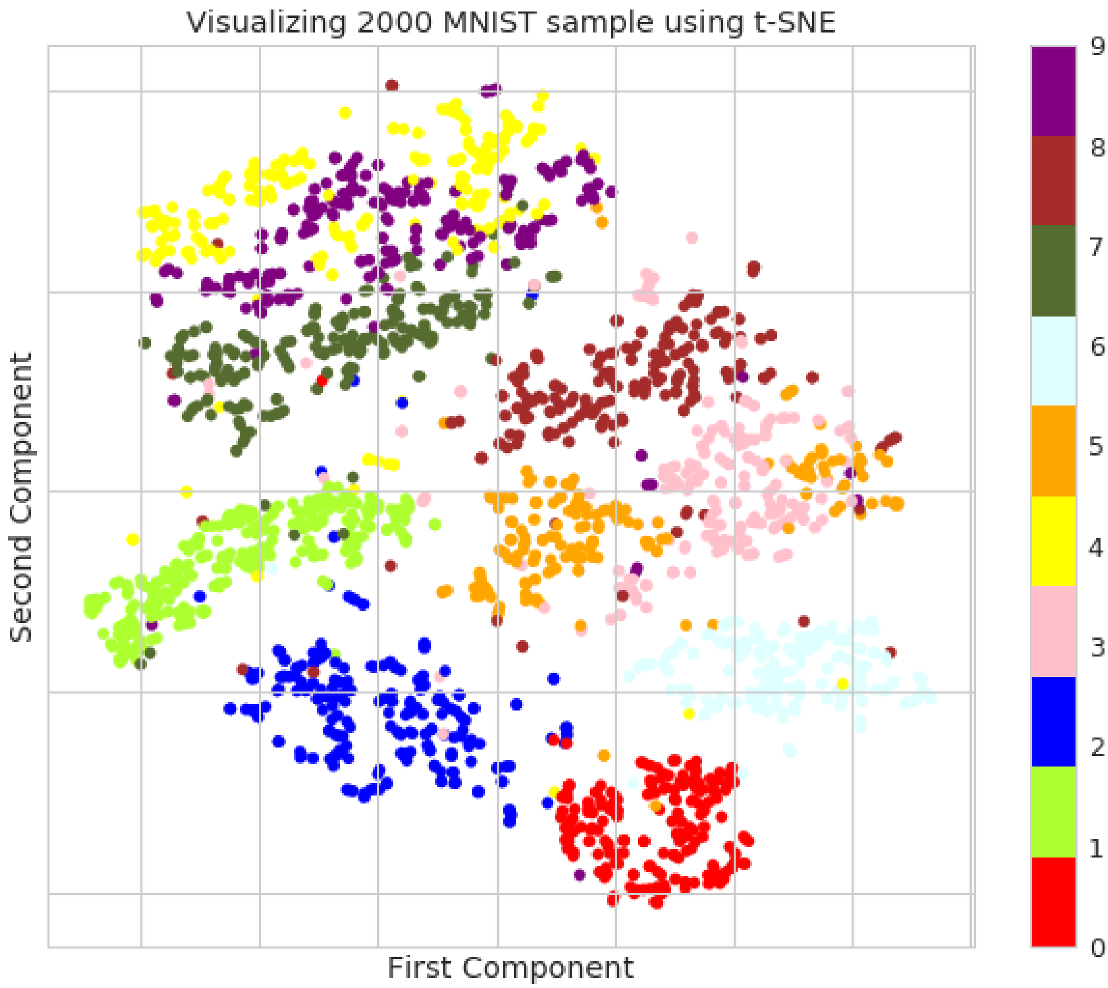
```
In [8]: from tsne import *
```

2.0.1 Two Dimensional t-SNE

```
In [ ]: #runs t-SNE
        #2 dimensions
        Y = tsne(X, 2, 50, 20.0);
```

```
In [31]: #Plots the result
         plt.scatter(Y[:,0], Y[:,1], c=labels, cmap=cmap)
         #plt.axis('off')
         plt.grid(True)
         plt.title('Visualizing 2000 MNIST sample using t-SNE')
         plt.xlabel('First Component')
         plt.ylabel('Second Component')
         plt.tick_params(         # changes apply to the x-axis
             which='both',       # both major and minor ticks are affected
             bottom='off',       # ticks along the bottom edge are off
             top='off',
             right = 'off',
```

```
        left = 'off',
        labeltop = 'off',
        labelleft = 'off',
        labelright = 'off',
        labelbottom='off')
    plt.colorbar()
    plt.show()
```

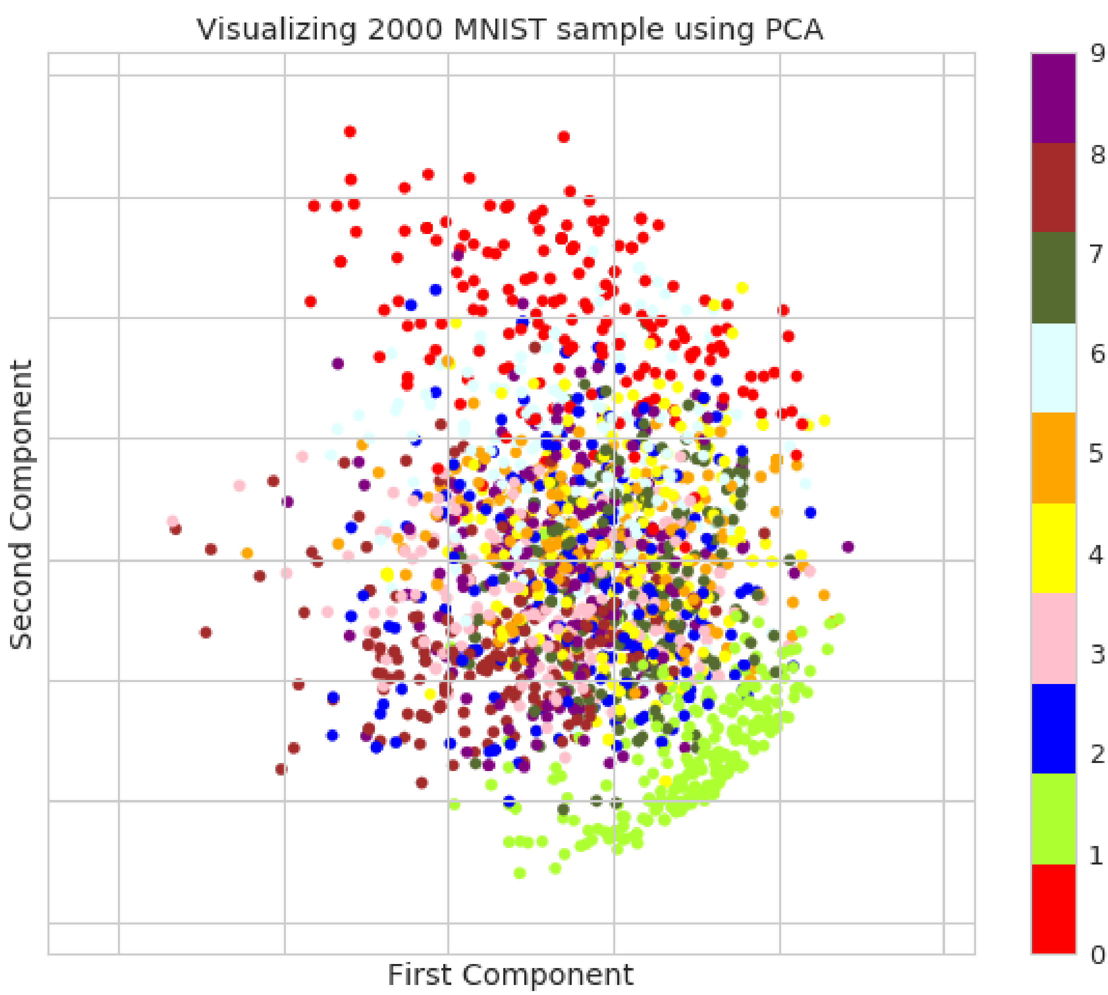


2.0.2 Compare with Two Dimensional PCA

```
In [24]: from sklearn.decomposition import PCA
         pca = PCA(n_components=2)
         pca.fit(X.T)

         Z = pca.components_.T
```

```
#Plots the result
plt.scatter(Z[:,0], Z[:,1], c=labels, cmap=cmap)
#plt.axis('off')
plt.grid(True)
plt.title('Visualizing 2000 MNIST sample using PCA')
plt.xlabel('First Component')
plt.ylabel('Second Component')
plt.tick_params(          # changes apply to the x-axis
    which='both',      # both major and minor ticks are affected
    bottom='off',      # ticks along the bottom edge are off
    top='off',
    right = 'off',
    left = 'off',
    labeltop = 'off',
    labelleft = 'off',
    labelright = 'off',
    labelbottom='off')
plt.colorbar()
plt.show()
```

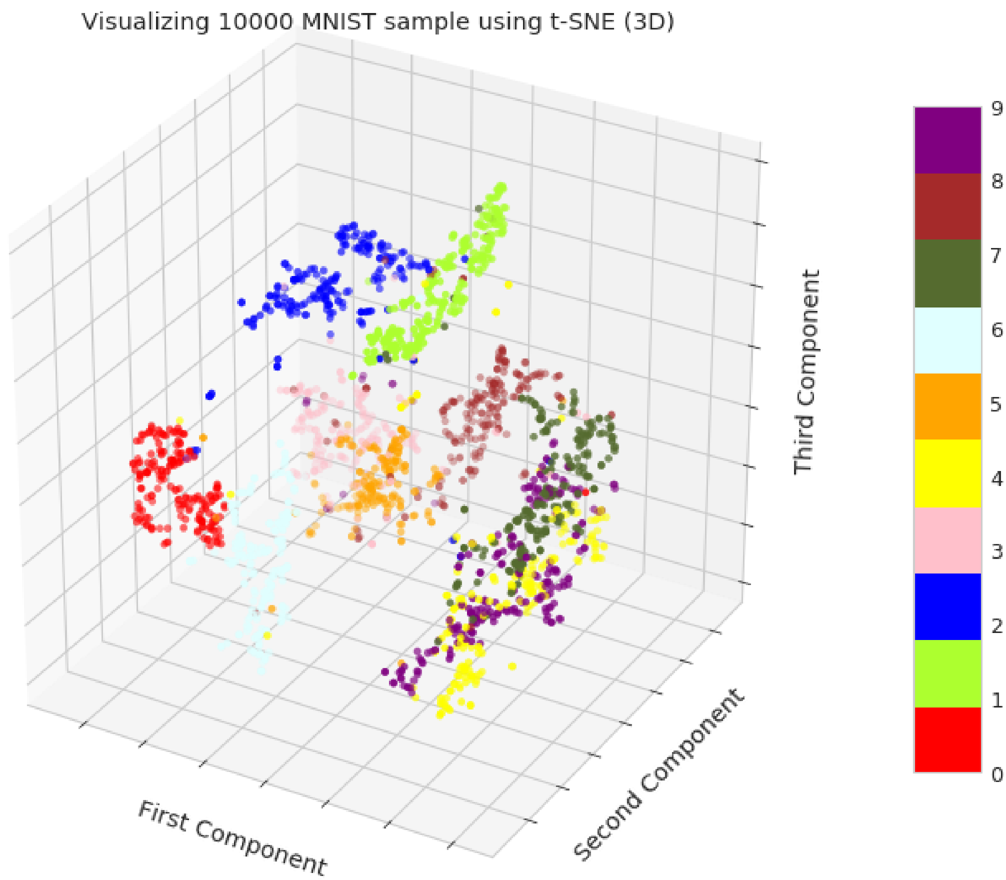


2.0.3 Three Dimensional t-SNE

```
In [ ]: #Now with three dimensions
        Y3 = tsne(X, 3, 50, 20.0);
```

```
In [25]: #Plots the result
         fig = plt.figure()
         ax = Axes3D(fig)
         sca = ax.scatter(Y3[:,0], Y3[:,1], Y3[:,2], c=labels, cmap=cmap)
         ax.set_title('Visualizing 10000 MNIST sample using t-SNE (3D)')
         ax.set_xlabel('First Component')
         ax.set_ylabel('Second Component')
         ax.set_zlabel('Third Component')
         ax.tick_params(         # changes apply to the x-axis
             which='both',      # both major and minor ticks are affected
             bottom='off',      # ticks along the bottom edge are off
             top='off',
             right = 'off',
             left = 'off',
             labeltop = 'off',
             labelleft = 'off',
             labelright = 'off',
             labelbottom='off')
         fig.colorbar(sca, shrink=0.7, aspect=10)

         plt.show()
```



2.0.4 Compare with Three Dimensional PCA

```
In [27]: pca = PCA(n_components=3)
         pca.fit(X.T)

         Z = pca.components_.T

         #Plots the result
         fig = plt.figure()
         ax = Axes3D(fig)
         sca = ax.scatter(Z[:,0], Z[:,1], Z[:,2], c=labels, cmap=cmap)
         ax.set_title('Visualizing 10000 MNIST sample using PCA (3D)')
         ax.set_xlabel('First Component')
         ax.set_ylabel('Second Component')
         ax.set_zlabel('Third Component')
         ax.tick_params(          # changes apply to the x-axis
             which='both',       # both major and minor ticks are affected
```

```
    bottom='off',      # ticks along the bottom edge are off
    top='off',
    right = 'off',
    left = 'off',
    labeltop = 'off',
    labelleft = 'off',
    labelright = 'off',
    labelbottom='off')
fig.colorbar(sca, shrink=0.7, aspect=10)

plt.show()
```