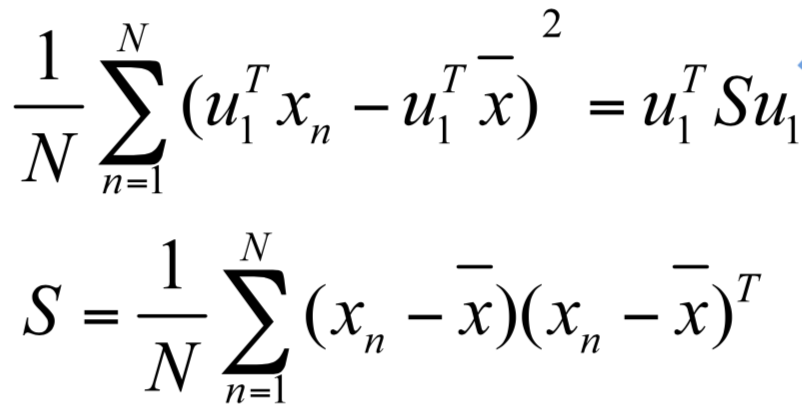
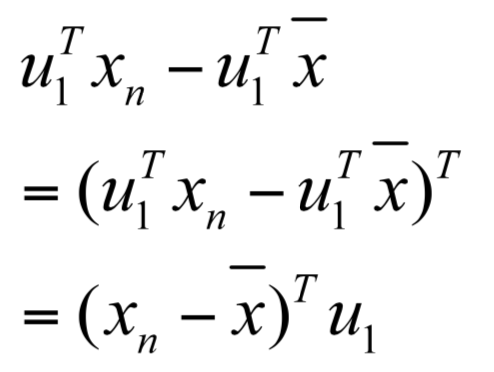
For this section, you will have a wonderful trip of understanding two dimensional reductions methods mathematically. And I will also introduce how to apply sklearn package to implement PCA and TSNE in projects. Next section, we will the coding parts of our projects on PCA and TSNE.

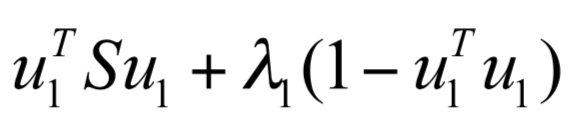
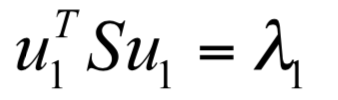
1. Mathematical trip for PCA and TSNE
2. PCA – principal component analysis.

Linear Projection method: projection maximizing the variance of the projected points.

We would first use one-dimensional space as an example

1. Construct the variance among all data points. S is the variance. Set be unit vector that represents the direction of one dimensional subspace that we want to project our data points on. In such cases, We represent he variance among all data points in the following way.

With following detailed steps, we get the expression of our euqation

1. In order to maximize the variance S for the case of one dimension, we need to use the Lagrange multiplier method
2. By setting the derivative respecting to equal to zero:
3. By multiplying on both sides, we then get the following equation, which represent the variance of the data-points along the new principal component’s direction. We choose as the new direction for lower space.

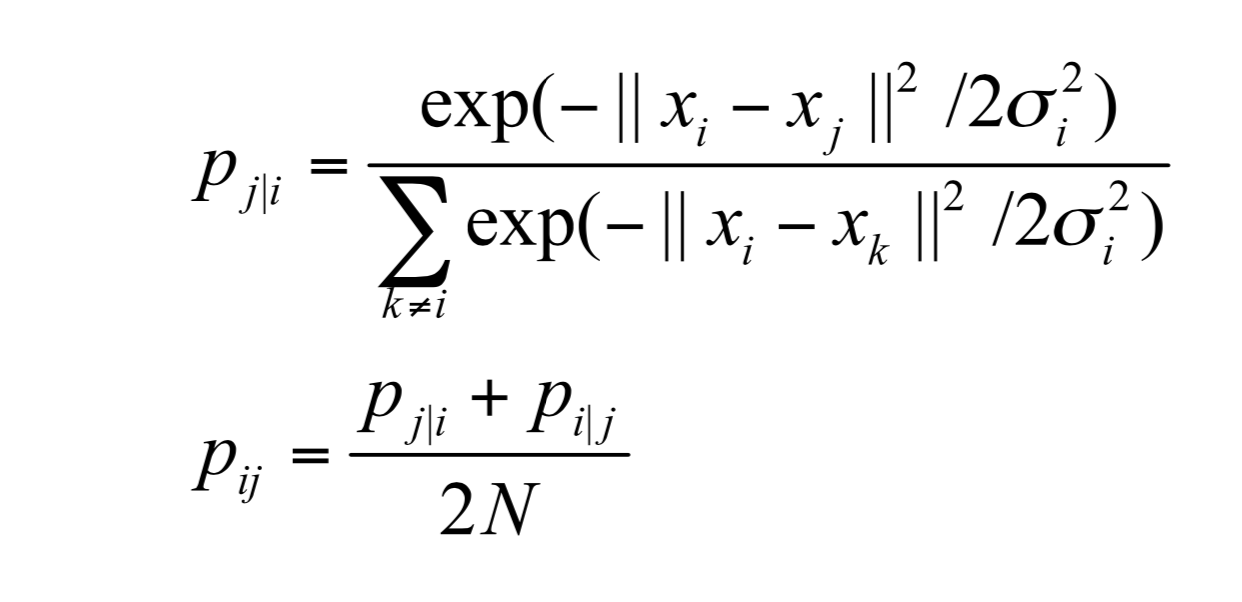
*\*Note: for the two eigen vectors, we choose the one with larger eigenvalues.*

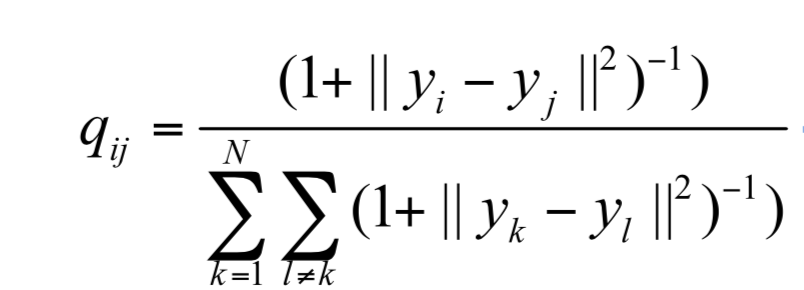
For the n-dimensional maximization, we just need to transform metric into n by n covariance matrixes. And we perform the maximization on all dimensions just as discussed previously using the same thoughts.

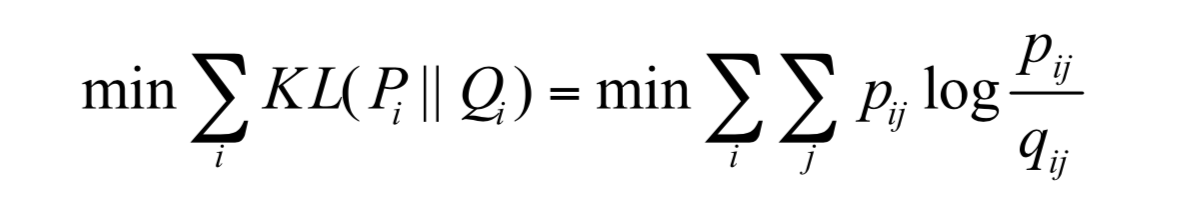
TSNE - t-Distributed Stochastic Neighbor Embedding

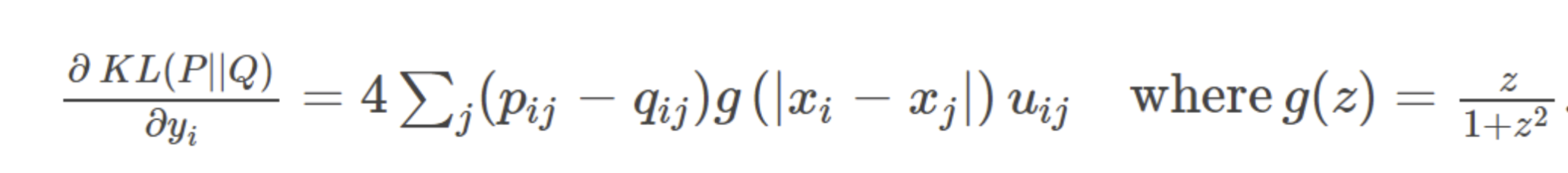
Using student-t distribution to model the lower dimensional space (assuming information is missing), which approximates the distribution of higher dimension. We need to minimize the divergence between two distributions.

1. Distribution (P) construction over pairs of the higher dimension data points. We use L2 norm to measure the distance between two data points. And the goal is that: similar objects have a high probability of being picked together, while dissimilar points have very low probability of being picked together.



1. Secondly, we construct the distribution (Q) of lower dimension using student-t distribution approximations (which is usually used to approximate normal distribution considering the scenario of missing information, like the sample distribution of normal distribution)
2. Thirdly, implement the Kullback-Leibler divergence equations between similarity measurement distribution (P) in high dimension and that (Q) in low dimension. In such cases, we just need to minimize the divergence value for the two distributions, and then the argument that minimizing the divergence would be the new data points in low dimensional spaces.



1. For minimization, we are using the gradient descent and update y ( for each iterations, until it is within the minimal error range.

Embedding: projection minimizing the divergence between original distribution and current distribution: the distribution that measures pairwise similarities of the input objects and a distribution measuring pairwise similarities of the corresponding low-dimensional points of the embedding.

The difference x

(2) Tutorial of PCA and TSNE implementation

Scikit-learn is a simple and efficient python tool for data analysis and machine learning. It is access to everyone and it can be reused to various contexts. It is open source and is built on matplotlib, Numpy and Scipy. Scikit-learn requires the following environments, please make sure you already have python, Numpy and SciPy installed.

* Python (>= 2.7 or >= 3.3),
* NumPy (>= 1.8.2),
* SciPy (>= 0.13.3).

If you already have Numpy and Scipy installed on your machine, do the following in the command window. Run one of the following options

Option 1:

pip install -U scikit-learn

Option 2:

(python2 -m pip install -U scikit-learn) **for python2**

python3 -m pip install -U scikit-learn) **for python3**

\*Note: *< python3/python2 -m> is used for specific python version you want to use for sklearn. You can link scikit-learn modules to either python2 or python3.*

If your machine does not have Numpy and Scipy on your computer, please do the following two commands first.

pip install -U scipy

pip install -U numpy

On mac OS, you can simply do

pip install -U numpy scipy scikit-learn

If you hate the terminal windows, but still want to have a simple interface. (You can use jupyter notebook, interactive python notebook) It is an interactive shell running python and it is very easy to use and view the results. It is exactly good for be

pip install –upgrade pip

pip install jupyter

Now let us discover how to apply PCA and TSNE in scikit-learn packet. Please open up a new python file (python2, python3, notebook or events the shell) We use jupyter notebook for the next section.

import numpy as np

import matplotlib.pyplot as plt

import matplotlib.colors

import pandas as pd

The reason I import matplotlib is because we can use matplot lib to visualize how data points are projected onto lower dimensional space.

Run the following command to start the notebook (option)

jupyter notebook

Open up the browser and direct to <http://localhost:8888/>, you can work with the jupyter notebook.

The sklearn.decomposition is a file under sklearn module that contains a function called PCA, which we would use for our projects. Thus, we do the following code to include sklearn.decomposition’s PCA (built-in dimension reduction function using PCA in sklearn package) function in our project, as well as the TSNE function (built-in dimension reduction function using TSNE) function in our project.

from sklearn.decomposition import PCA

from sklearn.manifold import TSNE

In next lines of the codes. We are generating 100 data points using numpy for us to perform the dimension reduction. The following codes are for generating those samples with two different labels, 0, and 1.

# generate 100 random data points with label 0 in 4-D, for N(2,4), label them as 0

data\_points\_label\_0 = 2\*np.random.randn(100,10)+2

# generate 100 random data points with label 1 in 4-D for N(4,9)

data\_points\_label\_1 = 3\*np.random.randn(100,10)+4

# generate 100 random data points with label 2 in 4-D for N(6,16)

data\_points\_label\_2 = 4\*np.random.randn(100,10)+6

In the following code, we generate the labels and color list for all data points.

#generate the label and color lists

label = ['0','1',’2’]

color = ['red','green','blue']

The following codes are used for dimension reduction using sklearn PCA and TSNE methods. We want to project the data from 10-dimensional space onto 2-dimensional space, which is better for visualization. The “n\_components” is the parameter for passing the dimension of the new vector space that you would like to transform your original data 2. There are other parameters that you can set up, like the number of iterations or the least error range (their default values are already good enough for most of our projects), in such case, we only define the dimension in our cases. If you want to use your own prefered values, you can view the official API for PCA and TSNE online. Those two functions are used to operate linear transformation on all data points, projecting them from higher into lower dimensional spaces.

#project data points onto 2-dimension space using PCA

data\_PCA = PCA(n\_components=2).fit\_transform(data\_points)

#project data points onto 2-dimension space using TSNE

data\_TSNE = TSNE(n\_components=2).fit\_transform(data\_points)

The following codes are using matplotlib.plot library to visualize the results of the PCA and TSNE for my labeled samples. You can see the tutorial of matplotlib online. Those codes are separating two graphs for both TSNE and PCA.

#then plotting the data points comparting PCA and TSNE

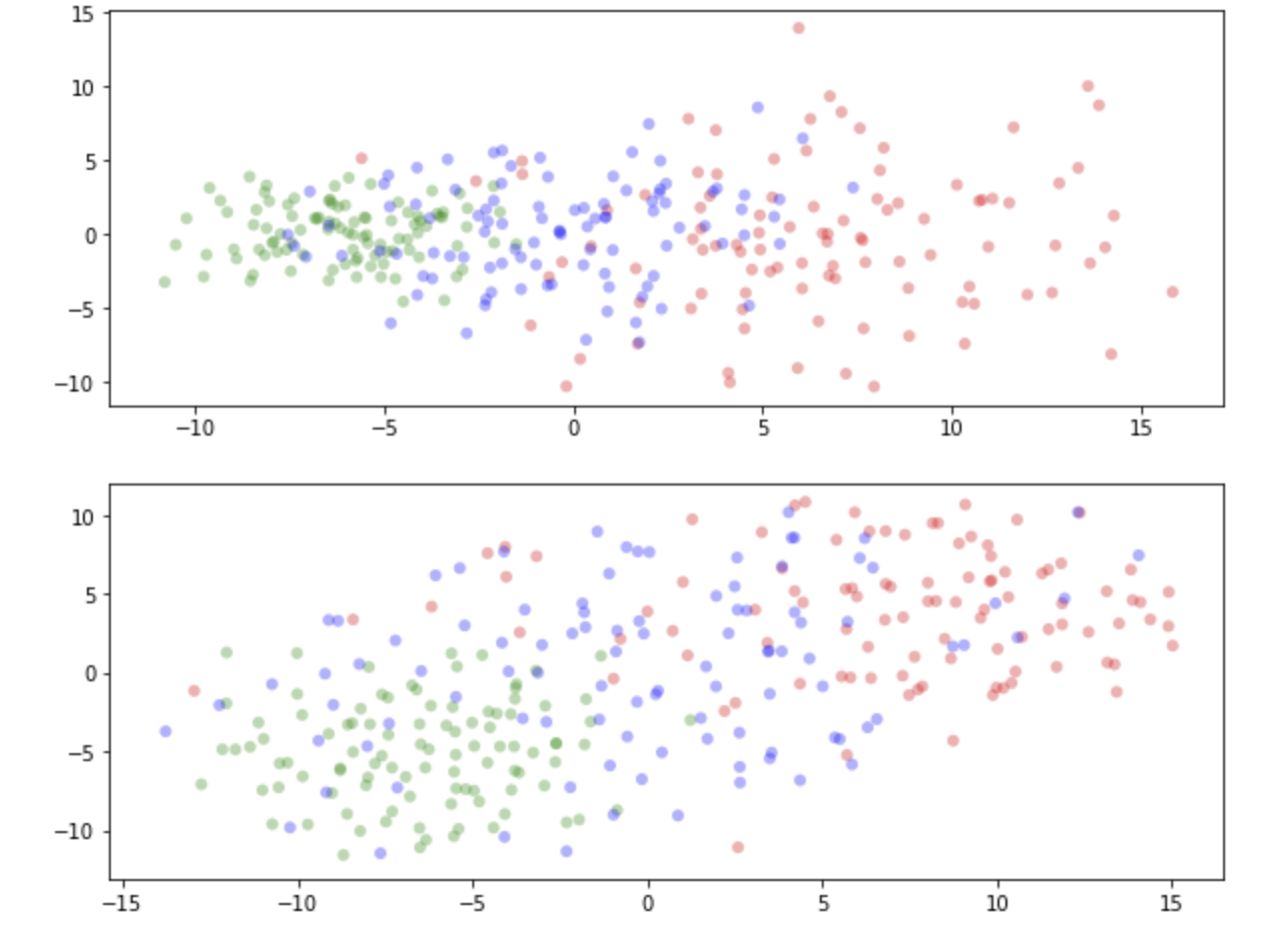
fig,ax = plt.subplots(2,1,figsize=(10,8))

for i in range(len(label)):

ax[0].scatter(data\_PCA[i\*100:(i+1)\*100,0],data\_PCA[i\*100:(i+1)\*100,1],c=color[i],label=label[i],alpha=0.3, edgecolors='none')

ax[1].scatter(data\_TSNE[i\*100:(i+1)\*100,0],data\_TSNE[i\*100:(i+1)\*100,1],c=color[i],label=label[i],alpha=0.3, edgecolors='none')

And there is a question for you guys, which one is result of TSNE and which one is that of PCA? Or it is hard to tell with our example? Or they work well for our ideal example in the tutorial



If you want to see a better example displaying the comparison between PCA and TSNE, please go to the following webpage. This is for the hand-written digits, you can clearly see the difference.

<https://xiangzhoujushi.github.io/application_materials/Project2/functionality_Extension.html>

Sum:

Comparing the TSNE and PCA, those two methods use different strategies. TSNE applies the covariance matrices and statistical approximation to scatter points and project points into lower dimension from Gaussian Distribution to small-samples T distribution. It is trying to preserve the neighborhood in lower dimension. In contrast, PCA uses the main component vectors. The first vector is the highest variance vector for first dimension we choose, and the second one is the highest variance vector for the second dimension we choose that is orthogonal to first one. Thus, points are related somehow through two directions along those two vectors. (in x and y in our visualizations). In such case, PCA does not preserve the relationship between relevant points.

In such case, for our manipulation to transform the dense vector to the sparse one, we are using TSNE instead of PCA, (as genre classifies our data), because it works better for preserve the clustering and that is pivotal information on the labeled data (relevant data).

And we also have a very high dimensional word vector, we want to preserve the relationship between data in such case.

Reference:

<https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding>

<http://scikit-learn.org/stable/developers/advanced_installation.html>