67750 | Advanced Practical Course in Machine Learning | Exercise 3

Guy Lutsker 207029448

Part I

Theoretical Part

Question 1

Show that $S = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})(x_i - \bar{x})^\intercal$ is PSD. assume $\bar{x} = 0$

$Solution$:

Firstly, let us recall that a matrix A is PSD if $\forall v$ it hold that $v^\intercal A v \geq 0$.

And so let us look at $v^\intercal S v = v^\intercal(\frac{1}{n-1}\sum_{i=1}^{n}(x_i)(x_i)^\intercal)v = \frac{1}{n-1}\sum_{i=1}^{n} v^\intercal(x_i)(x_i)^\intercal v$

$= \frac{1}{n-1}\sum_{i=1}^{n}(v^\intercal x_i)^2$ which is a sum of non negatives, and so S is PSD as required.

Question 2

Show that the data sits on a d-dimensional subspace $V \subset \mathbb{R}^n$ n if and only if S is of rank d. hint: show that S and X have the same rank.

$Solution$:

Notice that since the mean is zero, S can be expressed as $S = \frac{1}{n-1}\sum_{i=1}^{n} x_i x_i^\intercal = \frac{1}{n-1}\sum_{i=1}^{n} X^\intercal X$

And so S's rank is equivalent to $X^\intercal X$'s and so we just need to show that X has the same rank as $X^\intercal X$.

(sounds a lot like a problem we saw in IML last semester, where we needed to show that $Ker(X) = Ker(X^\intercal X)$)

Conjecture:

$$rank(X) = rank(X^\intercal X)$$

Proof 1:

$$rank(X) = dim(IM(X)) \overset{Dimension\ Theorem}{=} dim(\mathbb{R}^n) - dim(Ker(X))$$

$$\rightarrow rank(X) = rank(X^\intercal X) \Leftrightarrow dim(Ker(X)) = dim(Ker(X^\intercal X))$$

So I will show a harder problem which is that the kernels are actually equal $Ker(X^\intercal) = Ker(XX^\intercal)$:

Proof taken from my exercise in IML last semester:

Let $X \in \mathbb{R}^{pXn}$ be a matrix we will prove $Ker(X^T) \subset Ker(XX^T) \wedge Ker(XX^T) \subset Ker(X^T)$

$Ker(X^T) \subset Ker(XX^T)$:

Let $v \in Ker(X^T)$ then $XX^T v = X \cdot 0 = 0 \rightarrow v \in Ker(XX^T)$

$Ker(XX^T) \subset Ker(X^T)$:

let $v \in Ker(XX^\intercal) \rightarrow XX^\intercal v = 0$ lets look at $0 = v^\intercal XX^\intercal v =$

$(X^\intercal v)^\intercal (X^\intercal v) \rightarrow X^\intercal v \overset{vector\ that\ orthogonal\ to\ itself\ is\ the\ 0\ vector}{=} 0 \rightarrow v \in Ker(X^\intercal)$

Proof 2:

Let us use the Spectral theorem and rewrite X as $X = UDU^*$ where D is diagonal and U is unitary and particularly, for our application, its full rank. and so:

$$X = UDU^*$$

$$X^\intercal X \overset{Spectral\ decomposition\ of\ X}{=} (UDU^*)UDU^* = UDIDU^* = UD^2U^*$$

And so since U is unitary we can say that $rank(X) = rank(UDU^*) = rank(D)$

And since D is diagonal we know that its rank is the number of non zero column in the matrix, and D^2 has the same number of non zeros columns as D and so $rank(D) = rank(D^2)$.

And all together we get that:

$$rank(X) = rank(UDU^*) = rank(D) = rank(D^2) = rank(UD^2U^*) = rank(X^\intercal X)$$

And so :

$$rank(X) = rank(S)$$

Question 3

Show that the new coordinates are the result of an isometry on the subspace V .

$Solution$:

Recall that in PCA $S = UDU^\intercal$ and the new coordinates $\{y_i\}_{i=1}^n = \{U^\intercal x_i\}_{i=1}^n$

We need to show that $\forall i \ \|x_i\| = \|y_i\|$, then lets look at $\|y_i\|$:

$$\|y_i\| \overset{Defenition}{=} \|U^\intercal x_i\| \overset{\substack{x_i \text{ can be represented as} \\ \text{a linear combination of } U \\ \text{s.t: } x_i = \sum_{i=1}^n a_i u_i}}{=} \|U^\intercal \sum_{i=1}^n a_i u_i\| = \|\sum_{j=1}^n \sum_{i=1}^n a_i u_j^\intercal u_i\|$$

$$\overset{U \text{ is orthogonal}}{=} \|\sum_{j=1}^n \sum_{i=1}^n a_i u_i \delta_{i,j}\| = \|\sum_{i=1}^n a_i u_i\| = \|x_i\|$$

Question 4

In this question we will walk through the derivation of the second stage of LLE (finding the W matrix). In class, we saw that the goal of the second stage of LLE is to describe every point x_i as an affine combination of its k nearest neighbors:

$$W_{i,:} = \underset{w}{argmin}||x_i - \sum_{j\in N(i)} w_j x_j||^2 = \underset{w}{argmin}||\sum_{j\in N(i)} w_j z_j||^2$$

Where we define $z_j = x_j - x_i$.

Define G to be the Gram matrix of the z vectors, namely $G_{a,b} = z_a^T z_b$.

1. Show that $||\sum_{j\in N(i)} w_j z_j||^2 = w^T G w$.

$$Solution:$$

$$\|\sum_{j\in N(i)} w_j z_j\|^2 = \langle \sum_{j\in N(i)} w_j z_j, \sum_{j\in N(i)} w_j z_j\rangle = \sum_{i,j} w_i w_j z_i^\intercal z_j \overset{G_{i,j}=z_i^\intercal z_j}{=} \sum_{i,j} w_i w_j G_{i,j} \overset{by\ def}{=} w^\intercal G w$$

Question 5

2. The above means that we can find w by minimizing the quadratic form $w^T G w$, under the constraint that $\sum_i w_i = 1$. Formulate the Lagrangian and get the solution as a function of λ:

$$w = \frac{\lambda}{2} G^{-1}\mathbf{1}$$

$$Solution:$$

Our restrain is $\sum_i w_i = 1 \Leftrightarrow w^\intercal\ \mathbf{1} = 1 \Leftrightarrow w^\intercal \mathbf{1} - 1 = 0$,and so our function is $f(w,\lambda) = w^\intercal G w - \lambda(w^\intercal \mathbf{1} - 1)$.

deriving by w yield(remembering that G is symmetric):

$$\frac{\partial f}{\partial w} = w^\intercal G + w^\intercal G - \lambda \mathbf{1} = 2w^\intercal G - \lambda\mathbf{1}$$

And so finding the roots of this function:

$$2w^\intercal G - \lambda\mathbf{1} = 0 \Leftrightarrow w^\intercal = \frac{\lambda\mathbf{1}}{2G} = \frac{\lambda}{2} G^{-1}\mathbf{1}$$

and with respect to w instead of $w^\intercal$ we get:

$$w = (\frac{\lambda}{2} G^{-1}\mathbf{1})^\intercal = \frac{\lambda}{2} G^{-1}\mathbf{1}$$

since λ is a scalar, G is symmetric, and $\mathbf{1}$ is not effected by transpose either.

Question 6

Read the first four pages of "Diffusion Maps, Spectral clustering and Eigenfunctions of Flokker-Planck Operator" and repeat here the proof of the diffusion map theorem.

$$Solution:$$

Lets start with some definition:

We have our data X. The first thing we do is define a pairwise similarity matrix using the heat kernel with width ε:

$$L_{i,j} = exp(-\frac{\|x_i - x_j\|^2}{2\varepsilon})$$

Next we evaluate a diagonal normalization matrix $D_{i,i} = \sum_j L_{i,j}$

And a matrix M as a low dimensional representation of the data : $M = D^{-1}L$

Notice that M is adjoint to a symmetric matrix M_s : $M_s = D^{\frac{1}{2}}MD^{-\frac{1}{2}}$.

And so M, M_s have the same eigenvalues.

Denote $\{\lambda_j\}_{j=1}^{n-1}$ as those eigenvalues, and $\{v_j\}_{j=1}^{n-1}$ the eigenvectors of M_s,and ϕ_j, ψ_j as the left and right eigenvectors of M.

Notice that:

$$\forall j \in [n-1] : \phi_j = v_j D^{\frac{1}{2}},\ \psi_j = v_j D^{-\frac{1}{2}}$$

And since v_j are orthonormal to each other that means that: :

$$\forall i, j \in [n-1] : \langle \phi_j, \psi_j \rangle = \delta_{i,j}$$

Note that : $\phi_0(x_i) = \frac{D_{i,i}}{\sum_j D_{j,j}}$ with eigenvalue 1.

Now, lets define the probability for time t: $p(t, y|x) = \phi_0(y) + \sum_{j\geq 1} a_j(x)\lambda_j^t \phi)j(y)$

Where $a_j(x) = \psi_j(x)$.

And we would also like to quantify the similarity between any two points according to the evolution of their probability distributions. Specifically, we consider the following distance measure at time t:

Set $w(y) = \frac{1}{\phi_0(y)}$ and define:

$$D_t^2(x_0, x_1) = \|p(t, y|x_0) - p(t, y|x_1)\|_w^2 = \sum_y (p(t, y|x_0) - p(t, y|x_1))^2 w(y)$$

Next denote the mapping between the original space and the first k eigenvectors as the diffusion map:

$$\Psi_t(x) = (\lambda_1^t \psi_1(x), \lambda_2^t \psi_2(x), ..., \lambda_k^t \psi_k(x))$$

Now, FINALLY, to our main theorem:

$$Theorem:$$

$$D_t^2(x_0, x_1) = \sum_{j\geq 1} \lambda_j^{2t}(\psi_j(x_0) - \psi_j(x_1))^2 = \|\Psi_t(x_0) - \Psi_t(x_1)\|^2$$

$$Proof:$$

Since we know that $p(t, y|x) = \phi_0(y) + \sum_{j\geq 1} a_j(x)\lambda_j^t \phi)j(y)$

And that $D_t^2(x_0, x_1) = \sum_y (p(t, y|x_0) - p(t, y|x_1))^2 w(y)$

We can say that:

$$D_t^2(x_0, x_1) = \sum_y (\sum_j \lambda_j^t(\psi_j(x_0) - \psi_j(x_1))\phi_j(y))^2 \cdot \frac{1}{\phi_0(y)}$$

and from $\forall j \in [n-1] : \phi_j = v_j D^{\frac{1}{2}},\ \psi_j = v_j D^{-\frac{1}{2}}$ and from $\forall i, j \in [n-1] : \langle \phi_j, \psi_j \rangle = \delta_{i,j}$

We get that $\sum_y(\sum_j \lambda_j^t(\psi_j(x_0) - \psi_j(x_1))\phi_j(y))^2 \cdot \frac{1}{\phi_0(y)} = \|\Psi_t(x_0) - \Psi_t(x_1)\|^2$
As required.

Part II
Practical Part

1 Implementation of Manifold Learning Algorithms

1.1 Preface

In this part of the exercise I implemented several manifold learning algorithms: MDS, LLE, DM. We were asked to run our implementations of several data sets.

1.2 Random 2D data set

For this part, we were asked to generate 2D data, embed it in a higher dimension(i chose 1000 dimensions), rotate it in an arbitrary direction in the higher dimension and try to learn the manifold using MDS. But firstly lets look at the scree plot of the data :

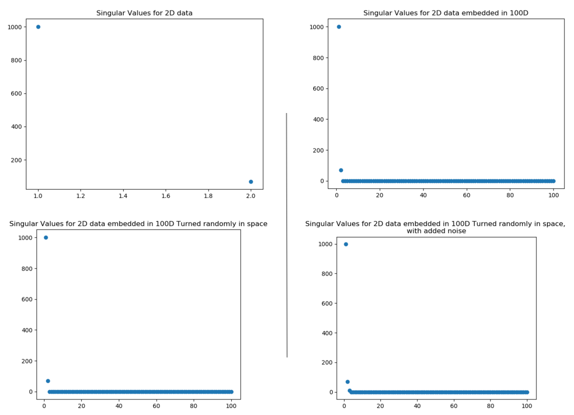

Figure 1: Scree plots for the data though all the steps

As we might have expected there are only 2 singular values which don't change even if we rotate them in a higher dimension(sanity check) and once we add noise there are some non zero singular values, but they are small if the noise is small - if the noise is larger than it is indistinguishable from the "real" singular value. The results for MDS in this data with Gaussian noise(mean 0, var 1) are shown in figure.2:

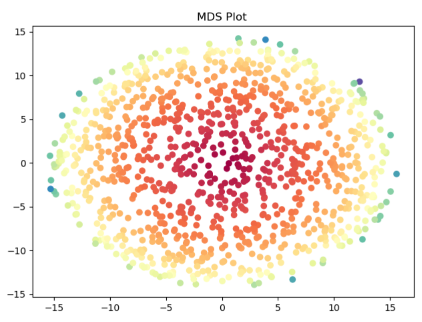

Figure 2: MDS plot on rand_2d_data (color is the original magnitude of the samples in 2D)

As we can see the data was restored to 2D space and the points seem to sit in 2D space rather comfortably. The original data magnitude in 2D space is shown in color and we can tell from that that the embedding preserved the original data's orientation and order - the color is continuous, as well as monotonous (in the outgoing direction of the origin) as we would have expected. In addition to evaluate the algorithms performance farther I ran the same data but plotted it in 3D space: https://www.cs.huji.ac.il/~guy_lutsker/2d_rand_data_MDS_on_3d_graph_gussian_noise.html . As we can see the data was restored to a 2D manifold rather beautifully - the data seems to sit on a paraboloid(2D object in 3D space) which once more confirms our embedding in 2D space. When playing around with the mean nothing significant seems to happen, but an interesting quirk that happens when we play around with the variance is that the data approaches a 1D line in space :

www.cs.huji.ac.il/~guy_lutsker/2d_rand_data_MDS_on_3d_graph_gussian_noise_with_high_var.html. Also when using random noise(and not Gaussian) the data approaches a saddle:

www.cs.huji.ac.il/~guy_lutsker/2d_randdata_MDS_on_3d_graph.html .

1.3 Swiss Roll Data Set

Let us compare the results of MDS,LLE, DM on the Swiss roll data set.

1.3.1 MDS

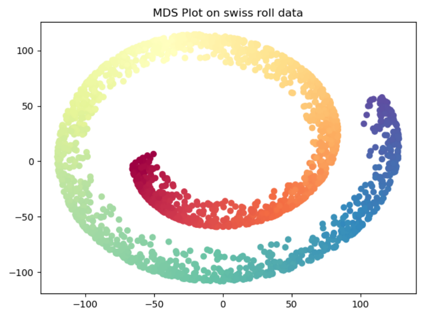

Figure 3: MDS plot on Swiss roll

As we can wee the algorithm failed on this data set as we were not able to capture the 2D manifold of the data.

1.3.2 LLE

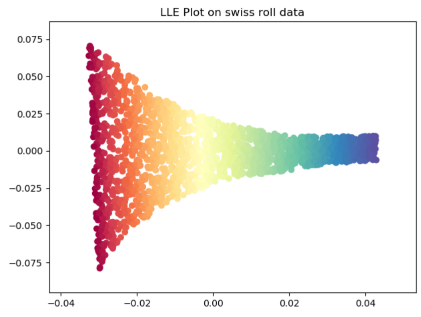

Figure 4: MDS plot on rand_2d_data (color is the original magnitude of the samples in 2D)

As we can wee the algorithm succeeded on this data set as we were able to capture the 2D manifold of the data - we got an "unrolling" of the Swiss roll in 2D.

3D plot : www.cs.huji.ac.il/~guy_lutsker/LLE_12n.html

1.3.3 Diffusion Map

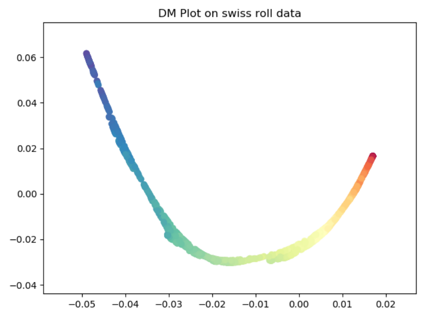

Figure 5: MDS plot on Swiss roll

As we can wee the algorithm failed on this data set as we were not able to capture the 2D manifold of the data.

1.4 Faces Data Set

1.4.1 MDS

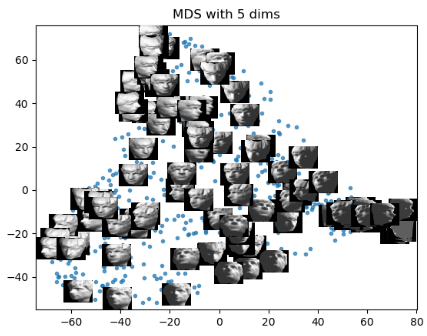

Figure 6: MDS plot on faces data

Here we an see that the algorithm was able to "catch" something. we can see that is a patch of dark images in the right corner which don't really go with any thing else in the visualization. But all other images behave rather well - we can see that faces were embedded in space with some correlation to their orientation in space. the left corner are faces looking to the left, and faces looking to the right are in the top. Overall it seems that this method succeeded.

1.4.2 LLE

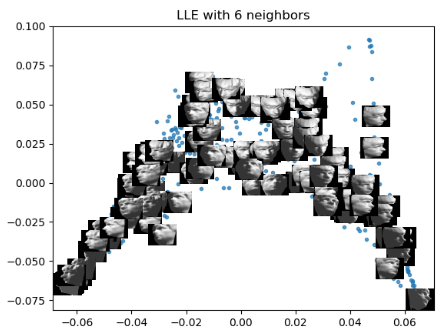

Figure 7: LLE plot on faces data

This is a very interesting result to my eyes. First of all, I think we can agree the algorithm worked as we wanted. Faces looking to the left are in the left of the plot, and faces looking to the right are in the right of the plot. But whats most interesting in my opinion, in the rather continuous transition we see from looking to the left, to looking to the right, meaning faces in the middle are looking with a duller angle. In addition faces which are not looking to each side, are clustered as well - faces looking down are in the middle up of the plot, and faces looking up are in the middle bottom of the plot.

1.4.3 Diffusion Map

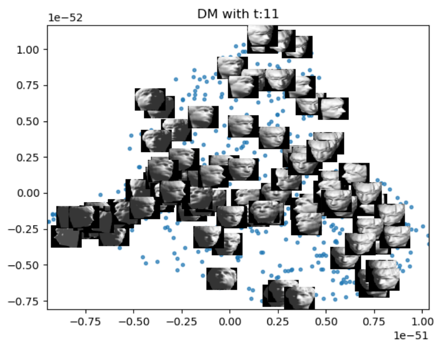

Figure 8: DM plot on faces data

As we can wee the algorithm kind of worked, as the dark patch is in the left of the plot, and faces are kind of clustered according to their orientation(less well then the others). To get this I had to check a lot (A LOT) of options, and only an extreme value to sigma(1000 - 2000) worked for me.

1.4.4 Conclusion

It seemed that all method "caught" two major features in the images data set - orientation of the face(its angle) and the lighting effects in the image, but altogether it seems like all 3 algorithms did rather well.

Part III

Netflix Prize Dataset(NPD) - Practical Manifold Learning Research

2 Preface

In this section we were asked to participate in an actual manifold learning task from the real world on the Netflix prize dataset, using the tools we learned in class.

3 Preprocessing

To preprocess the Netflix Prize Dataset I used Matan's code which was published in the piazza website. To be totally honest, my first 3 days of the assignment were spent doing exactly what Matan did, only he had found a file in google with the genres of the movies, and I tried to mine that data myself from IMDB which took me forever... And as it usually happens in life, I found that someone else already did what I tried and did it much better than I could have. And so I decided to use Matan's code for preprocessing.

4 Exploring the DataSet

4.1 Embedding in Lower Linear Subspace

The first thing I wanted to to is to reduce the dimensionality of the data so I could start manipulating it. Running the data though a linear reduction like PCA sounded like a good idea at first, but when giving it a bit of research online, it looked a though my data would not benefit from a standard PCA reduction. The data consists of a very sparse matrix in an extremely high dimension and from what I understood a good approach for this kind of data is replacing PCA with TruncatedSVD. From the Algorithms documentation: "This transformer performs linear dimensionality reduction by means of truncated singular value decomposition (SVD). Contrary to PCA, this estimator does not center the data before computing the singular value decomposition. This means it can work with sparse matrices efficiently".

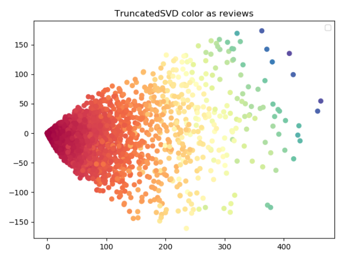

Figure 9: TruncatedSVD on NPD)

Figure 8. shows how TruncatedSVD displaces the data onto 2 dimensions. I wanted to understand the 2 dimensions TruncatedSVD "chose" and so based on this graph we can see that the first dimension correlates pretty well with how well the movie did in terms of review scores (no additional intuition comes in 3D plot).

4.2 Exploring Non-Linear Dimension Reductions

Next I wanted to see if I could explore the higher manifold of this data, in order to do this we have to leave the comfortable linear subspaces and turn to non linear approaches.

4.2.1 MDS

The manifold learning algorithm I tried running was the MDS algorithm. I ran it on the reduced data(50 linear dimensions):

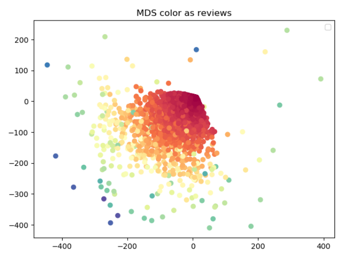

Figure 10: MDS on NPD, color as review scores

The major conclusion I can say about MDS is that it looks a lot like PCA/TruncatedSVD results - it looks like it explodes from the origin and correlated with review scores. In addition they both look like a funnel in 3D space.

4.2.2 LLE

Next I tried running LLE on the data set. I tried multiple parameters and got the following results:

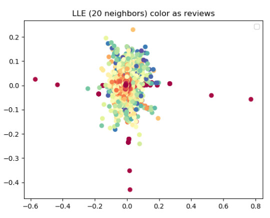

Figure 11: LLE on NPD, color as review scores

I tried running this method with many different parameters(number of neighbors, number of components,looking at different components,etc..) and non seemed to show a meaningful result, I thought about not including this plot, but I spent so long trying to tune the parameters to see something at that it seemed like a waste.

4.2.3 Diffusion Map

Another non linear approach we already saw is diffusion maps. I tried running the algorithm on the reduced data(50 linear dimensions) set with different parameters and got the following results:

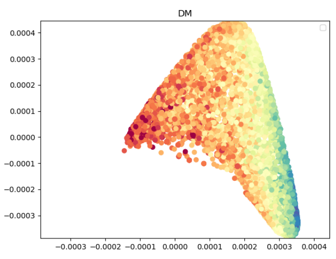

Figure 12: DM on NPD, color as review scores

Figure 9. shows us the results of DM of the dataset, and we can see that algorithm was able to capture some features in the data. For example we can once more see that the first dimension of the DM is correlated with review scores. And it is also interesting to see the 3D plot of the DM: www.cs.huji.ac.il/~guy_lutsker/DM_3d_by_rev_logged.html

. And colored by genera: www.cs.huji.ac.il/~guy_lutsker/DM_3d_by_gen.html . Firstly the 3D graph here is very interesting since the data in the 3D DM seems to sit on a 2D plane(non linear though, and so it will not appear well on a 2D DM) And so it might be interesting to run 2D LLE on top of the 3D DM to see how the manifold behaves(Guy from the future here, unfortunately it didn't work...) . Secondly we can again see that review score is highly impacting the manifolds shape, but movie genera(which I always looked at and tried to color by it) is, sadly, not correlated to any axis in the DM.

4.2.4 t-SNE

The first actual non linear dimension reduction I tried was t-SNE. t-SNE is a tool to visualize high-dimensional data. It converts similarities between data points to joint probabilities. The results:

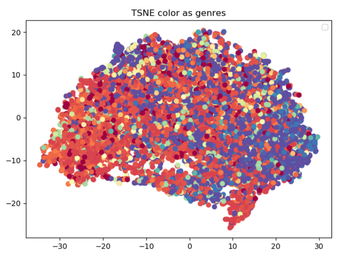

Figure 13: t-SNE on NPD, color as genres

This time I decided to show the coloring against movie genres(since the coloring against reviews looked like all others, and wasn't really interesting) and this time we actually see some coloring according to movie genres! It's not entirely obvious and its hard to say whether this is conclusive(will try to check in clustering analysis further) but in each method I tried to look at movie genres and here its the first time that some genres seemed to cluster together.

4.2.5 UMAP

Another non linear dimension reduction I tried was UMAP, UMAP is a relatively modern approach to non linear dimension reduction, and this method tried to preserve the topology of the data in the higher dimension(in more recent research UMAP starts to overtake t-SNE). My results with UMAP:

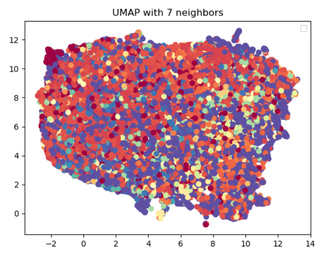

Figure 14: UMAP on NPD, color as genres

This results seems to be very similar to t-SNE, there isn't a major conclusion here besides that it has(maybe?) some correlation with movie genres. Although I have to say I'm pretty disappointing since I expected this method to be victorious, and that I would see something interesting in this visualization. I feel like I have to emphasize that I gave this method a lot of effort and generated around ~200 graphs on UMAP attempts alone, just to try and find good parameters to run it with with no luck :(

4.3 Clustering

The next intuitive step was trying to cluster the data. Since I didn't know which visualization represented the data I decided to look at TruncatedSVD, TSNE, DM. I decided to cluster to 20 clusters because of 2 reasons: 1. In my heart I really wanted the data to cluster according to the movie genres, and since there were ~28 movie genres in the data, a number of clusters around this number was attractive. 2. I decided to look at visualizations of all 3 methods in any numbers of clusters between 3 - 30 and above 20 the clusters were not really changing too much, and the visualization seemed well enough for my purpose in 20 clusters, and so I decided to go with 20. A note about this issue: I am aware that my reasoning here is not perfect, in particular choosing the number of clusters based on the visualization alone could be a terrible idea since the visualization always lies to us(we just never how). But since I could spend weeks on this issue alone, that this line of reasoning is good enough.

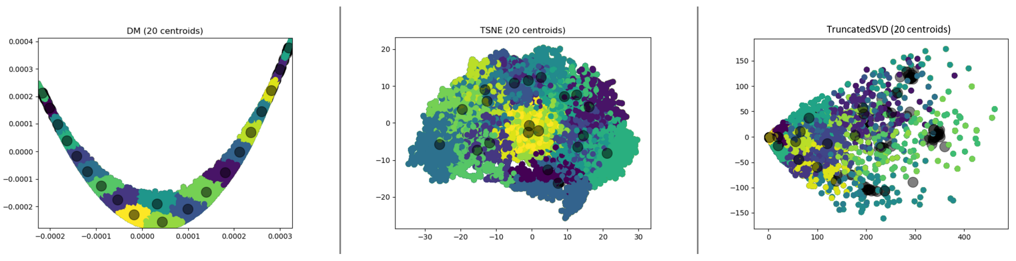

Figure 15: Clustering the NPD using KMeans

As we can see its hard to reach any meaningful conclusion from this clustering visualization. My hopes for clustering according to movie genres weren't really met, to say the least. The only interesting part of this Visualization is the left most cluster in the TSNE plot - it seems to be correlated with orange cluster(actually movie genre) in Figure 13.

4.4 Discussion & Conclusions

4.4.1 My Work

This was a very interesting(yet hard) exercise in manifold learning, we were given a data set with no further guidance and were asked to "research" it somehow. It was a really large data set and so it was very hard to handle it - opening it on my computer nearly crashed it, and every subsequent time froze my computer for the duration of the run(even trying to work on it though the university's cluster didn't really work). Though out my work I really wanted to find a reduction in dimension that will "cluster" the movies by genera, and I was really disappointed when no method really obtained the resulted I was hoping for. Perhaps running the algorithms with more samples(I only ran 50,000 samples) will yield better results or maybe better preprocessing/filtration of the data will help. Another thing I tried was checking whether the release date of the movie correlated with some axis of visualization and I checked it each time and never really saw something worth showing/noting :(On a personal note, despite that this exercise was very difficult for me, I work in a lab(Dr. Naomi Habib, Computational Neuroscience) that works with Single-Cell RNA data and it was very close the research I do in my work, and so I enjoyed the exercise :) .(Future Guy here again, its the day of the submission, and I was just informed that Spectral clustering of scikit doesn't exactly do diffusion map, although its very similar. and so I'm sorry for the misdirection, I ran Spectral clustering, not DM)

4.4.2 Further Research

I feel like if I had more time I would have tried several other methods for exploring the manifold of this data set. I would have separated the data into genera to begin with, and then tried to learn each manifold separately - maybe something interesting would have come up in one of MDS,DM,t-SNE,UMAP etc... In addition I would have loved to try More clustering algorithms(perhaps even soft assignment ones) on the data to see how it will behave, but I didn't really have an intuition for other types of clustering methods, and I didn't want to just run other ones I am not familiar with just for more graphs. Another branch of thinking that I gave a lot of effort was tying to run some NLP algorithm on this data set, I will explain - I had two major ideas here: 1. running an LDA Topic modeling on top of the movie titles to see if it cluster(soft cluster according to topic assignment) according to genres. 2. running a Grade Of Membership algorithm on top of the sparse data matrix to see how the data will rearrange it self - perhaps a more continuous approach will yield better results. Unfortunately both ideas didn't really work(I only ever ran these kind of algorithms in R language and for some reason running them in python gave my trouble). I probably could have made it work but since I am a only a Bachelor student, I didn't really have the time to make it work(although I did gave this exercise a lot of effort), but I thought it was an interesting idea to mention it at least.