# APML - Ex 4

Rhea Chowers, 204150643

January 23, 2020

## Information Bottleneck

### Information

1. By definition:

$$H(X,Y) = -\sum_x \sum_y p(x,y) \cdot \log(p(x,y))$$

and

$$H(Y|X) = \sum_x p(x) H(Y|X=x) = -\sum_x p(x) \sum_y p(y|x) \cdot \log(p(y|x))$$

Expanding the first phrase using the fact that: $p(y,x) = p(y|x)\,p(x)$

$$H(X,Y) = -\sum_x \sum_y p(x,y) \cdot \log(p(x,y)) =$$

$$= -\sum_x \sum_y p(x,y) \cdot \log(p(y|x)\,p(x)) =$$

$$= -\sum_x \sum_y p(x,y) \cdot (\log p(y|x) + \log p(x)) =$$

$$= -\sum_x \sum_y p(x,y) \log p(x) - \sum_x \sum_y p(x,y) \log p(y|x)$$

Now since $\sum_y p(x,y) = p(x)$:

$$= -\sum_x p(x) \log p(x) - \sum_x \sum_y p(y|x)\,p(x) \log p(y|x) =$$

$$= H(X) - \sum_x p(x) \sum_y p(y|x) \log p(y|x) = H(X) + H(Y|X)$$

as required.

2. By definition:

$$CE\left(p,q\right)=-\sum_{x}p\left(x\right)\cdot\log q\left(x\right)=-\sum_{x}p\left(x\right)\cdot\left(\log q\left(x\right)-\log p\left(x\right)+\log p\left(x\right)\right)=$$

$$=-\sum_{x}p\left(x\right)\cdot\left(\log q\left(x\right)-\log p\left(x\right)\right)-\sum_{x}p\left(x\right)\log p\left(x\right)=-\sum_{x}p\left(x\right)\log\frac{q\left(x\right)}{p\left(x\right)}+H\left(X\right)=$$

$$=D_{K,L}\left(p,q\right)+H\left(X\right)$$

as required.

3. By definition:

$$I\left(X,Y\right)=\sum_{x,y}p\left(x,y\right)\cdot\log\frac{p\left(x,y\right)}{p\left(x\right)p\left(y\right)}=\sum_{x,y}p\left(x,y\right)\cdot\left(\log\frac{p\left(x,y\right)}{p\left(y\right)}-\log p\left(x\right)\right)=$$

$$=\sum_{x,y}p\left(x,y\right)\log p\left(x|y\right)-\sum_{x,y}p\left(x,y\right)\log p\left(x\right)=H\left(X\right)-\left(-\sum_{x,y}p\left(x,y\right)\log p\left(x|y\right)\right)=$$

$$=H\left(X\right)-\left(-\sum_{y}\sum_{x}p\left(x|y\right)p\left(y\right)\log p\left(x|y\right)\right)=$$

$$=H\left(X\right)-\left(-\sum_{y}p\left(y\right)\sum_{x}p\left(x|y\right)\log p\left(x|y\right)\right)=H\left(X\right)-H\left(X|Y\right)$$

as required.

## Statistics

1. The mean of the sample:

$$\overline{X}=\frac{1}{|X|}\sum_{X_i\in X}X_i$$

Assume the samples are drawn from a bernouli distribution. Therefore:

$$P_\theta\left(X_i=1\right)=\theta\,,\,P_\theta\left(X_i=0\right)=1-\theta$$

$$\Rightarrow P_\theta\left(X_i=x\right)=\theta^x\left(1-\theta\right)^{1-x}$$

Therefore for an entire sample:

$$P_\theta\left(X_1=x_1,X_2=x_2,...,X_n=x_n\right)=\prod_{i=1}^{n}\theta^{x_i}\left(1-\theta\right)^{1-x_i}$$

$$=\theta^{\sum x_i}\left(1-\theta\right)^{\sum 1-x_i}=\theta^{\sum x_i}\left(1-\theta\right)^{|X|-\sum x_i}=\theta^{|X|\cdot\overline{X}}\left(1-\theta\right)^{|X|-|X|\cdot\overline{X}}$$

$$\Rightarrow P_\theta\left(X_1=x_1,X_2=x_2,...,X_n=x_n|\overline{X}=\overline{x}\right)=$$

$$= \frac{P_\theta\left(\overline{X}=\overline{x}|X_1=x_1,X_2=x_2,...,X_n=x_n\right)P_\theta\left(X_1=x_1,X_2=x_2,...,X_n=x_n\right)}{P\left(\overline{X}=\overline{x}\right)} =$$

$$= \frac{\theta^{|X|\cdot\overline{x}}\left(1-\theta\right)^{|X|-|X|\cdot\overline{x}}}{P\left(\overline{X}=\overline{x}\right)}$$

Now since we are dealing with a Bernoulli distribution, $\overline{X} = \frac{1}{|X|}\sum_{X_i\in X} X_i = \frac{1}{|X|}\cdot \#\text{of } X_i = 1$, since if $X_i = 0$ it doesn't contribute to the mean. Therefore $P\left(\overline{X}=\overline{x}\right) = P\left(|X|\cdot\overline{x} = \#ofones\right) = \binom{|X|}{|X|\cdot\overline{x}}\theta^{|X|\cdot\overline{x}}\left(1-\theta\right)^{|X|-|X|\cdot\overline{x}}$ since we choose $|X|\cdot\overline{x}$ tosses to have the value 1 and the rest get the value 0. Therefore:

$$P_\theta\left(X_1=x_1,X_2=x_2,...,X_n=x_n|\overline{X}=\overline{x}\right) = \frac{\theta^{|X|\cdot\overline{x}}\left(1-\theta\right)^{|X|-|X|\cdot\overline{x}}}{P\left(\overline{X}=\overline{x}\right)} =$$

$$= \frac{\theta^{|X|\cdot\overline{x}}\left(1-\theta\right)^{|X|-|X|\cdot\overline{x}}}{\binom{|X|}{|X|\cdot\overline{x}}\theta^{|X|\cdot\overline{x}}\left(1-\theta\right)^{|X|-|X|\cdot\overline{x}}} = \frac{1}{\binom{|X|}{|X|\cdot\overline{x}}}$$

and this is independent of $\theta$, therefore the mean is a sufficient statistic.

2. We wish to show that the statistic $T = X_1$ is not minimal, meaning $P_\theta\left(X_1=x_1,X_2=x_2,...,X_n=x_n|X_1=x\right)$ is not independent of $\theta$. This is of course logical since all the $X$'s are drawn independently and therefore $X_1 \perp X_i$ for $i \neq 1$, meaning $X_1$ can't contribute information about any other $X_i$. Formally:

$$P_\theta\left(X_1=x_1,X_2=x_2,...,X_n=x_n|X_1=x\right) = \frac{\theta^{\sum x_i}\left(1-\theta\right)^{\sum 1-x_i}}{P\left(X_1=x\right)} = \frac{\theta^{\sum x_i}\left(1-\theta\right)^{\sum 1-x_i}}{\theta^{x}\left(1-\theta\right)^{1-x}}$$

$$= \theta^{(\sum x_i)-x}\left(1-\theta\right)^{(\sum 1-x_i)-(1-x)} = \theta^{\sum_{i\neq 1} x_i}\left(1-\theta\right)^{\sum_{i\neq 1} 1-x_i}$$

meaning the conditional probability is dependent on $\theta$.

3. Let $T\left(X_1,X_2,...,X_n\right) = \left(X_1,X_2,...,X_n\right)$ the statistic that returns the sample. Denote the sample $S$:

$$P_\theta\left(X_1=x_1,X_2=x_2,...,X_n=x_n|T\left(S\right)=S\right) =$$

$$P_\theta\left(X_1=x_1,X_2=x_2,...,X_n=x_n|X_1=x_1,X_2=x_2,...,X_n=x_n\right) = 1 \perp \theta$$

4. Let $S,T$ be any two minimal sufficient statistics. Therefore, by definition both $S,T$ are sufficient statistics. Since $S$ is minimal, by definition $S = S\left(T\right)$, on the other hand $T$ is minimal so $T = T\left(S\right)$. Therefore, given $S$ or $T$ we can calculate the other, and therefore neither contributes more information than the other on a sample, and therefore they are equivalent.

5. A minimal sufficient statistic $M$ is a function of any sufficient statistic $T$, and a sufficient statistic is a function of the sample $X$, therefore $X \to T \to M$ is a markov chain and from the data processing inequality we get that $I(T,X) \geq I(M,X)$ for any sufficient statistic $T$. Therefore a minimal sufficient statistic maintains $I(M,X) = \min_T I(T,X)$ since a minimal sufficient statistic is also a sufficient statistic.

## Information Bottleneck

1. The sampling process described can be presented as the following markov chain, since $Y$ is dependent on some parameter $\theta$:

   $$\theta \to Y \to X \to T$$

   We wish to show that $I(Y,X) = I(\theta,T)$. Since $T$ is sufficient we know from definition that $I(\theta,T) = I(\theta,X)$. From the markov chain we know that $X \perp \theta | Y \Rightarrow Y$ is a sufficient statistic of $X$ as well, meaning that $I(\theta,X) = I(\theta,Y)$. Therefore $I(\theta,T) = I(\theta,Y)$. From the data processing inequality we know that $I(Y,X) \leq I(Y,T)$ but since $T$ is a sufficient statistic we know that $X \perp \theta | T$ and therefore $\theta \to Y \to T \to X$ is also a markov chain and we get that $I(Y,X) = I(Y,T)$. The following is also a markov chain: $T \to \theta \to Y \to X$ and therefore from the data processing inequality $I(T,\theta) \geq I(T,Y)$ but from the original markov chain we know that $I(Y,T) \geq I(\theta,T)$ and combining the inequalities we get $I(Y,T) = I(\theta,T)$. But we've shown that $I(Y,X) = I(Y,T)$ and therefore we get that $I(Y,X) = I(\theta,T)$ as required.

2. Since a minimal sufficient statistic $T$ is a function of any other sufficient statistic, and for any sufficient statistic $S$, $X \perp \theta | S$ wecan therefore interpert this in terms of the data processing inequality and get that $Y \to T \to S \to X$ is a markov chain, therefore from the data processing inequality:

   $$I(T,Y) \geq I(S,Y)$$

   Since this is true for every sufficient statistic (and since we have proven that all minimal sufficient statistics are equivalent) we get that a minimal sufficient statistic maximizes $I(T,Y)$ with respect to any sufficient statistic. Proving that a minimal sufficient statistic $T$ maintains that $I(T,X) = \min_S I(S,X)$ for any sufficient statistic $S$ was already proven earlier.
   Concluding, a minimal sufficient statistic maintains that $\forall S$ a sufficient statistic:

   $$I(X,T) = \min_S I(S,X)$$

   $$I(T,Y) \geq I(S,Y)$$

   proving the required.

# Manifold Learning

## PCA

1. Let $S = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})(x_i - \overline{x})^T = \frac{1}{n-1}X^TX$ for $X$ the normalized sample matrix. Let $y \in \mathbb{R}^n$ be some vector:

$$y^TSy = y^T\left(\frac{1}{n-1}XX^T\right)y =$$

$$= \frac{1}{n-1}\left(y^TX\right)\left(X^Ty\right) = \frac{1}{n-1}\left\langle X^Ty, X^Ty\right\rangle = \frac{1}{n-1}\left|\left|X^Ty\right|\right|_2 \geq 0$$

We didn't assume anything about $y$, therefore this is true for any vector in $\mathbb{R}^n$, and by definition we get that $S$ is PSD matrix.

2. Since normalizing $X$ is subtracting a constant row vector $\overline{x}$ from each row, and since $\overline{x} = \frac{1}{n}\sum x_i \Rightarrow \overline{x} \in row(X)$, we get that for the normalized sample matrix $\overline{X}$: $row(X) = row\left(\overline{X}\right)$ (their row spaces are the same).
If the data sits on a d-dimensional subspace then that means that $rank(X) = dim(row(X)) = d \Rightarrow rank\left(\overline{X}\right) = dim\left(row\left(\overline{X}\right)\right) = d$, since there can be at most $d$ linearly independent samples.
Now we'll claim $rank\left(X^TX\right) = rank(X)$ and conclude that the data sits on a d-dimensional subspace iff $S$ is of rank $d$. We'll show that using the rank nullity theorem, meaning that $nullity\left(X^TX\right) = nullity(X) \Leftrightarrow rank(X) = rank\left(X^TX\right)$.
$\Leftarrow$Let $x \in nullity(X) \Rightarrow Xx = \mathbf{0} \Rightarrow X^TXx = X^T\mathbf{0} = 0 \Rightarrow x \in nullity\left(X^TX\right)$
$\Rightarrow$Let $x \in nullity\left(X^TX\right)$ s.t. $x \neq \mathbf{0}$. Then: $X^TXx = \mathbf{0} \Rightarrow x^TX^TXx = x^T\mathbf{0} = 0$. But on the other hand: $x^TX^TXx = \langle Xx, Xx\rangle = ||Xx||_2 = 0$ therefore $Xx = \mathbf{0} \Rightarrow x \in nullity(X)$ proving both directions.
We conclude, $rank(S) = rank(X)$, and $rank(X) = d$ iff the data lies on a d-dimensional subspace, therefore proving the required.

3. $XX^T$ is symmetric and orthogonaly diagonizable, and therefore $XX^T = U\Lambda U^T$. $S$ is $X$ but normalized and centered. The normalization of $XX^T$ is the same as dividing the eigenvalues on the diagonal of $\Lambda$ by $n-1$, and recentering is identical to observing the data in a new point of reference (or moving the origin), and therefore doesn't affect the distances between any two points and we'll ignore this. The data lies on the space spanned by the eigenvectors of $XX^T$ which are the columns of $U$. Since the rank is $d$, we know that the data points can be spanned using only $d$ eigenvectors of $XX^T$, meaning $\forall x \in X : x = \sum_{i=1}^{d}\langle x, u_i\rangle u_i$. Performing PCA spans the data on the d-dimensional subspace $V \subset \mathbb{R}^n$ using the eigenvectors of $S$ which are those defined by $U$. Therefore, after the dimensionality reduction, every point $x$ is still spanned by $u_1, ..., u_d$ and therefore distances

between points are preserved:

$$\underbrace{||x_k - x_j||}_{\text{distance before reduction}} = ||\sum_{i=1}^{n} \langle x_k, u_i \rangle u_i - \sum_{i=1}^{n} \langle x_j, u_i \rangle u_i|| \underbrace{=}_{\text{matrix of rank d}}$$

$$= ||\sum_{i=1}^{d} \langle x_k, u_i \rangle u_i - \sum_{i=1}^{d} \langle x_j, u_i \rangle u_i|| = \underbrace{||x_k^d - x_j^d||}_{\text{distance after reduction}}$$

and therefore the transformation is an isometry.

## LLE

1. For $G_{ij} = z_i^T z_j$:

$$||\sum_{j \in N(i)} w_j z_j||^2 = \left\langle \sum_{j \in N(i)} w_j z_j, \sum_{j \in N(i)} w_j z_j \right\rangle =$$

$$= \langle w_1 z_1 + w_2 z_2 + ... + w_n z_n, w_1 z_1 + w_2 z_2 + ... + w_n z_n \rangle =$$

$$= \sum_{i,j} w_i w_j z_i^T z_j = \sum_{i,j} w_i w_j G_{i,j} = w^T G w$$

as required.

2. We want $\sum w_i = 1 \Leftrightarrow w^T \mathbf{1} = 1 \Leftrightarrow w^T \mathbf{1} - 1 = 0$. Using lagrange multipliers:

$$f(w, \lambda) = w^T G w - \lambda \left( w^T \mathbf{1} - 1 \right) \Rightarrow$$

$$\frac{\partial f}{\partial w} = \frac{\partial}{\partial w} \left( w^T G w \right) - \lambda \mathbf{1} = w^T G^T + w^T G - \lambda \mathbf{1} \underbrace{=}_{G^T = G} 2 w^T G - \lambda \mathbf{1}$$

$$\Rightarrow \frac{\partial f}{\partial w} = 0 \Rightarrow 2 w^T G - \lambda \mathbf{1} = 0 \Rightarrow w^T = \frac{\lambda \mathbf{1}^T}{2} G^{-1}$$

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

as required.

## Diffusion Maps

1. Let $A_{ij} = P(X_t = x_j | X_{t-1} = x_i)$. We'll prove that $A_{ij}^t = P(X_t = x_j | X_0 = x_i)$ by induction:

   (a) For $t = 1$ this is trivial since it is just the definition of $A$:

$$A_{ij}^1 = A_{ij} = P(X_t = x_j | X_{t-1} = x_i) = P(X_1 = x_j | X_0 = x_i)$$

   as required.

(b) Hypothesis: assume correctness for $t$ - $A_{ij}^t = P\left(X_t = x_j | X_0 = x_i\right)$.

(c) Prove for $t+1$: Look at $A^{t+1} = A^t A$, and examine the i,j coordinate:

$$A_{ij}^{t+1} = \left(A^t A\right)_{ij} = \sum_k A_{ik}^t A_{kj} = \sum_k P\left(X_t = x_k | X_0 = x_i\right) \cdot P\left(X_{t'} = x_j | X_{t'-1} = x_k\right)$$

examine one part of the sum: $P\left(X_t = x_k | X_0 = x_i\right) \cdot P\left(X_{t'} = x_j | X_{t'-1} = x_k\right)$ - this is the probability of getting from $x_i$ to $x_k$ in $t$ steps times the probability of getting from $x_k$ to $x_j$ in one step. Therefore this is the probability of getting from $x_i$ to $x_j$ in $t+1$ steps, using $x_k$ at the $t$'th step. Therefore, summing over all $x_k$'s is the total probability of getting from $x_i$ to $x_j$ in $t+1$ steps (since the transition has to occur through some $x_k$). Notice that this is exactly the sum! therefore:

$$A_{ij}^{t+1} = \sum_k P\left(X_t = x_k | X_0 = x_i\right) \cdot P\left(X_{t'} = x_j | X_{t'-1} = x_k\right) = P\left(X_{t+1} = x_j | X_0 = x_i\right)$$

as required.

2. $A$'s rows are all non-negative and sum to 1. Therefore:

$$\forall i : \left(A\mathbf{1}\right)_i = \left\langle row_i^A, \mathbf{1} \right\rangle = \sum_j A_{ij} \cdot 1 = 1$$

$$\Rightarrow A\mathbf{1} = 1 \cdot \mathbf{1}$$

and we get that $\mathbf{1}$ is an eigenvector of $A$ with eigenvalue 1.

3. Assume by negation that $\exists \lambda$ s.t. $|\lambda| > 1 \wedge \exists u : Au = \lambda u$. Let $|u_i| = \max_k |u_k|$ the largest element of the corresponding eigenvector in absolute value. Therefore:

$$\left(Au\right)_i = \sum_j A_{ij} u_j = \lambda u_i$$

We know that $\forall i, j : 0 \leq A_{ij} \leq 1$ since it is a stochastic matrix. Therefore:

$$\left|\left(Au\right)_i\right| = \left|\sum_j A_{ij} u_j\right| \leq \sum_j A_{ij} \max_k |u_k| = \sum_j A_{ij} |u_i| = |u_i|$$

On the other hand:

$$\left|\left(Au\right)_i\right| = |\lambda u_i| = |\lambda| \cdot |u_i| > |u_i|$$

and we get that:

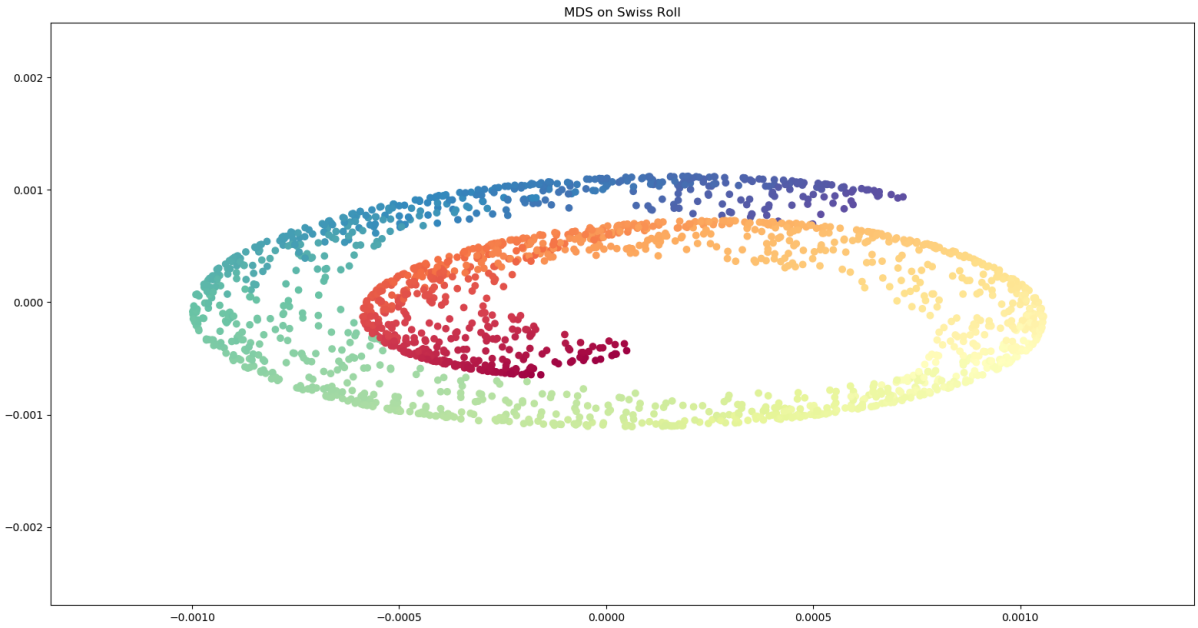
$$|u_i| < \left|\left(Au\right)_i\right| \leq |u_i|$$

which is a contradiction.

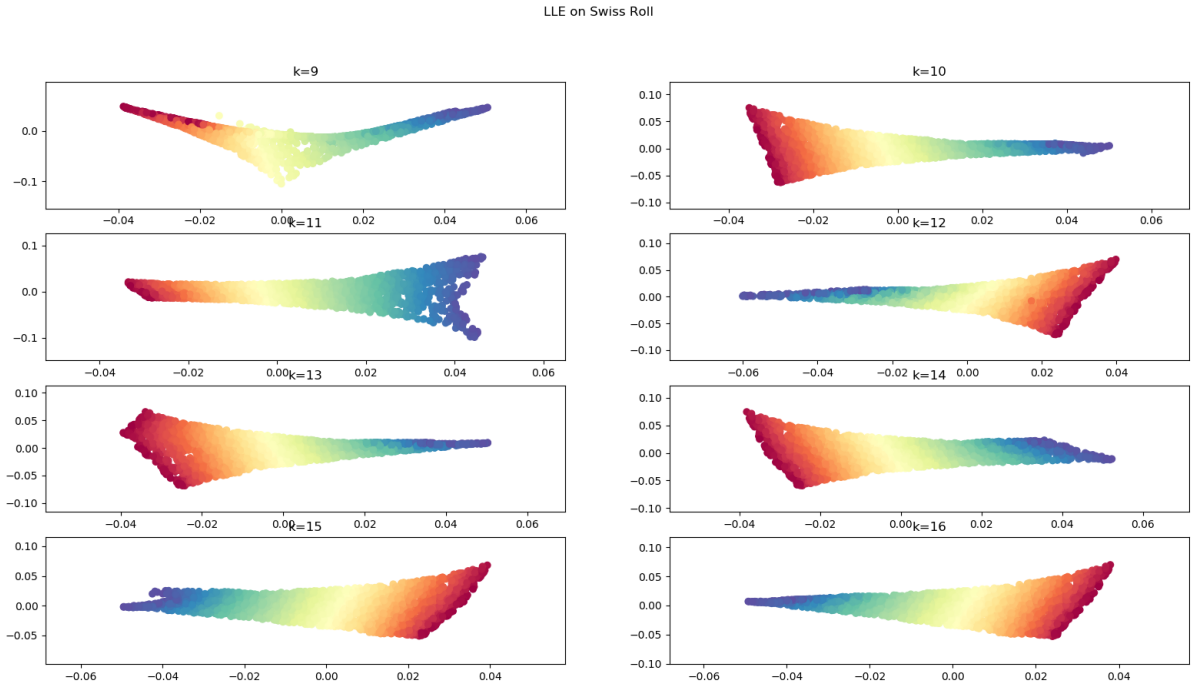
# Practical Part

## Swiss Roll Comparison

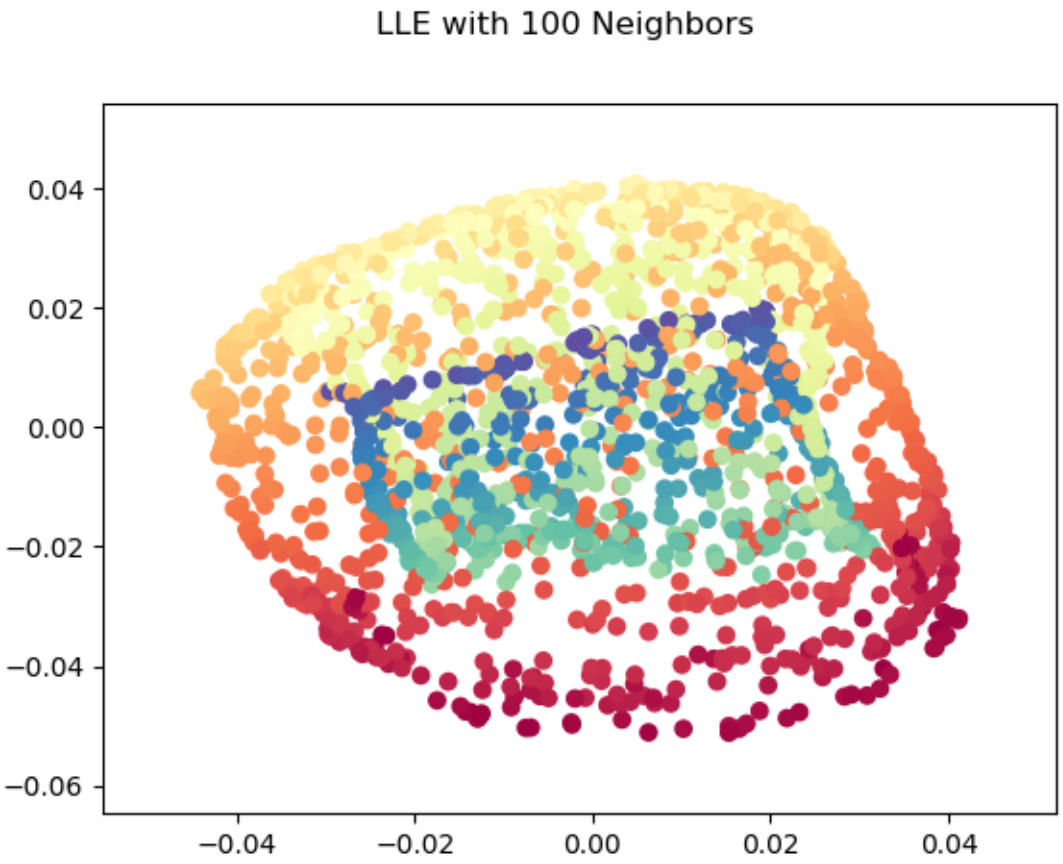
Let's compare the swiss roll's dimensionality reduction.



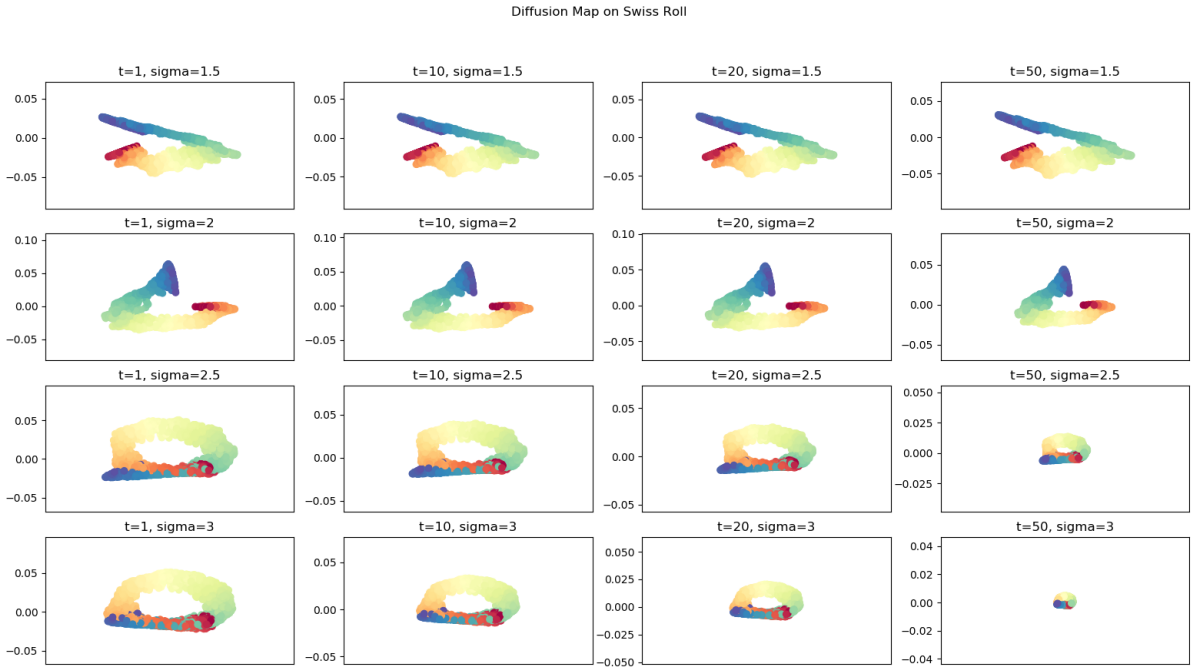

For MDS we can see that this is more or less a simple projection onto a linear subspace ("a slice of the swiss roll") which doesn't learn anything about the manifold itself. MDS is based on distances between points, but for the swiss roll, two points can be really close in the embedded dimension by far away in the intrinsic coordinates, for example two points which are on different "layers" of the swiss roll. We can see that this problem is preserved in 2-dimension and red points are close to green points (for example) although they are distant in the manifold's intrinsic coordinates. Concluding - MDS performs poorly on the swiss roll, and probably on any manifold with similar properties.




We can see that LLE performs pretty good, and the performance increases with the number of neighbors in the presented range.




Above we see what happens when increasing k too much. We rely on the assumption that the manifold is locally linear, and locality is defined by some $\delta$ which is a distance parameter between the point and its neighbors. Increasing $k$ is similar to taking a large neighborhood or $\delta$, and at that range the data is no longer linear, and therefore the reduction doesn't work as well.
In the next figure we mainly see that the value of $\sigma$ has an important effect on the dimensionality reduction using DM. This parameter, like $k$ in LLE defines the neighborhood of a point. We can see that at larger values of $\sigma$, the neighborhood for each point is larger and points in different layers of the swiss roll are considered neighbors, and we get that structure in 2-d.



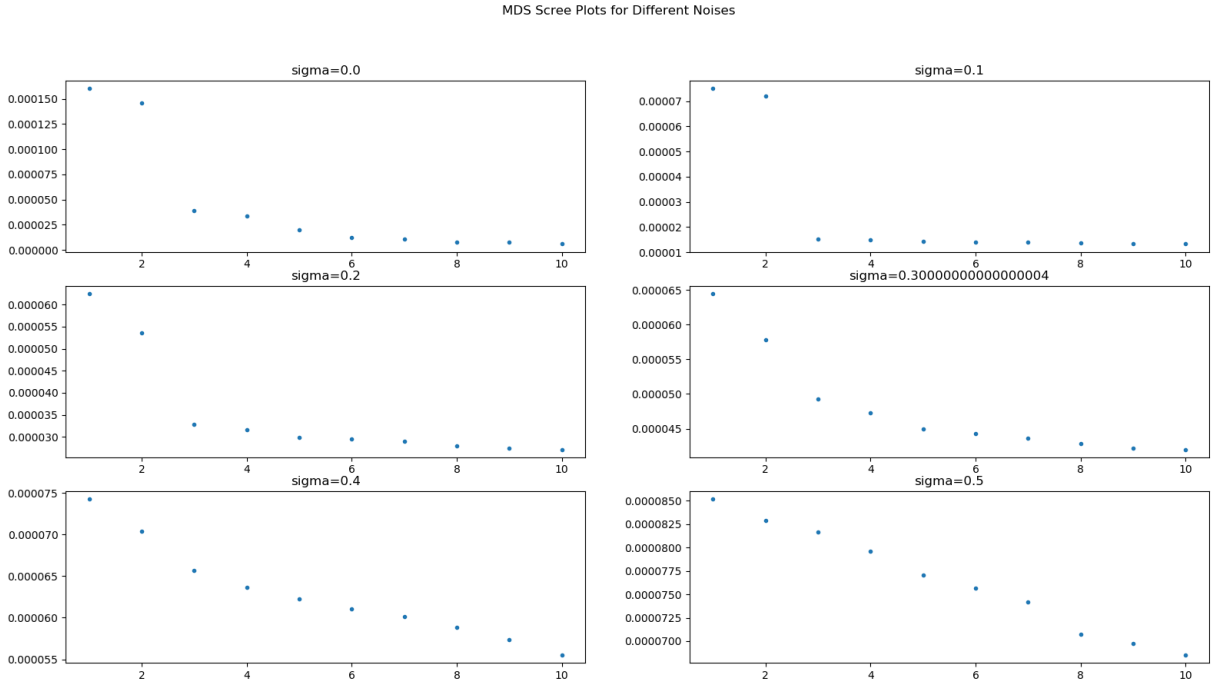

Concluding, MDS performs poorly on the swiss roll since it perserves distances. Since the swiss roll is symmetric along one of its axises, a 2-d isometry is simply slicing along the roll, which is exactly what we can see that MDS does. LLE performs good on the data, and finding

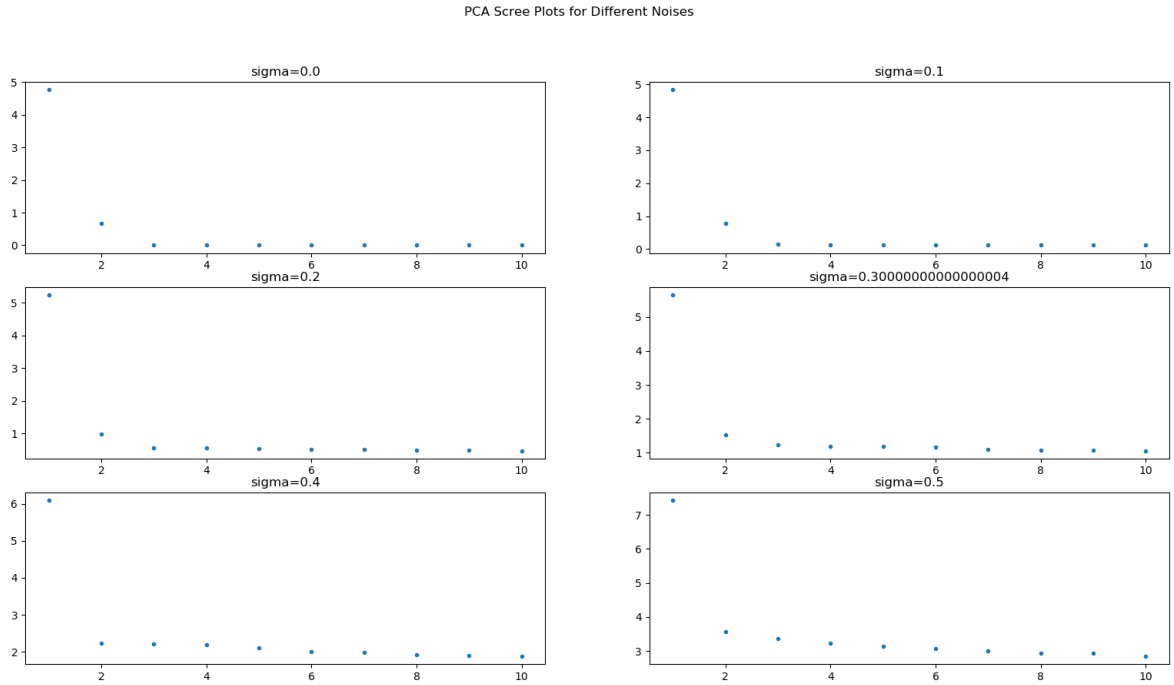
## Parameter Tweaking

As seen in the example above, LLE is easier to tweak relative to Diffusion Maps. Both these methods have a parameter which determines the neighborhood of a data point - the number of neighbors for LLE and $\sigma$ for DM. Both of them can be found using a "zoom in" search - I searched around a large range of parameters, and then focused around one which returned a good result in order to tweak it. On the other hand, DM have another parameter $t$ which determines the diffusion process. Since $t$ is used as a power of the eigenvalues of a stochastic matrix, taking large values of t (above 100) causes numerical issues, and therefore limits. In the swiss roll example, we can see that this parameter doesn't really affect the shape of the reduced data, and is more useful in clustering (since taking a large $t$ reduces the probability of "jumping" between far points, and therefore gives better seperation).

## Scree plots

Look at the scree plots generated by MDS and PCA for different noises (=values of $\sigma$ when sampling noise from a gaussian distribution).



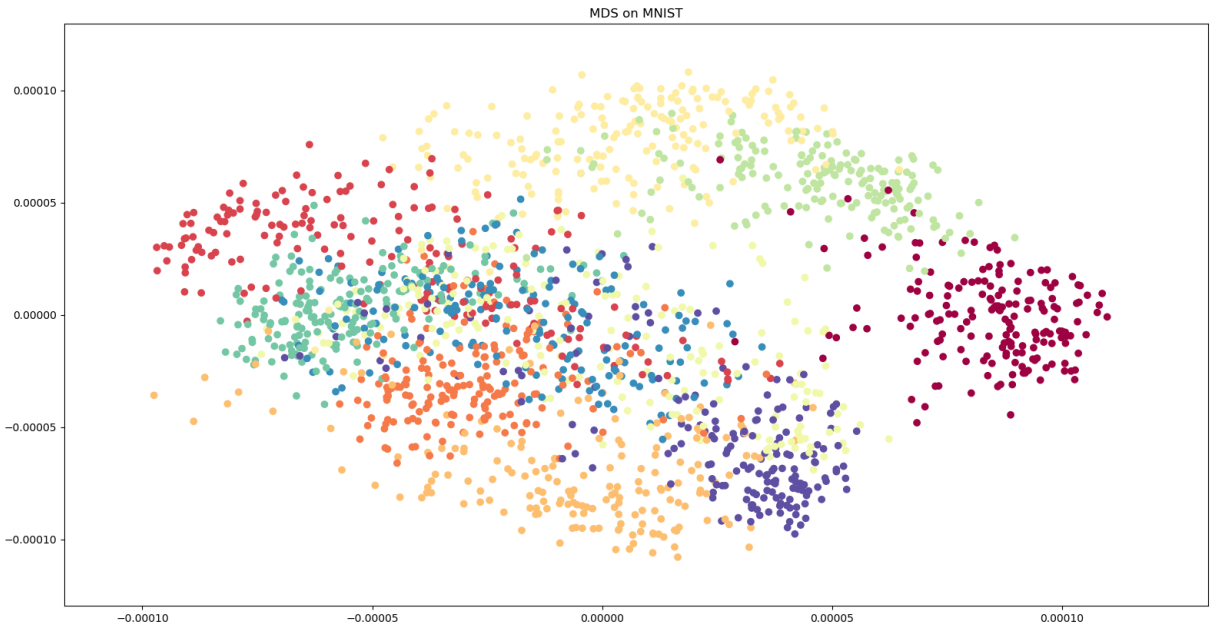


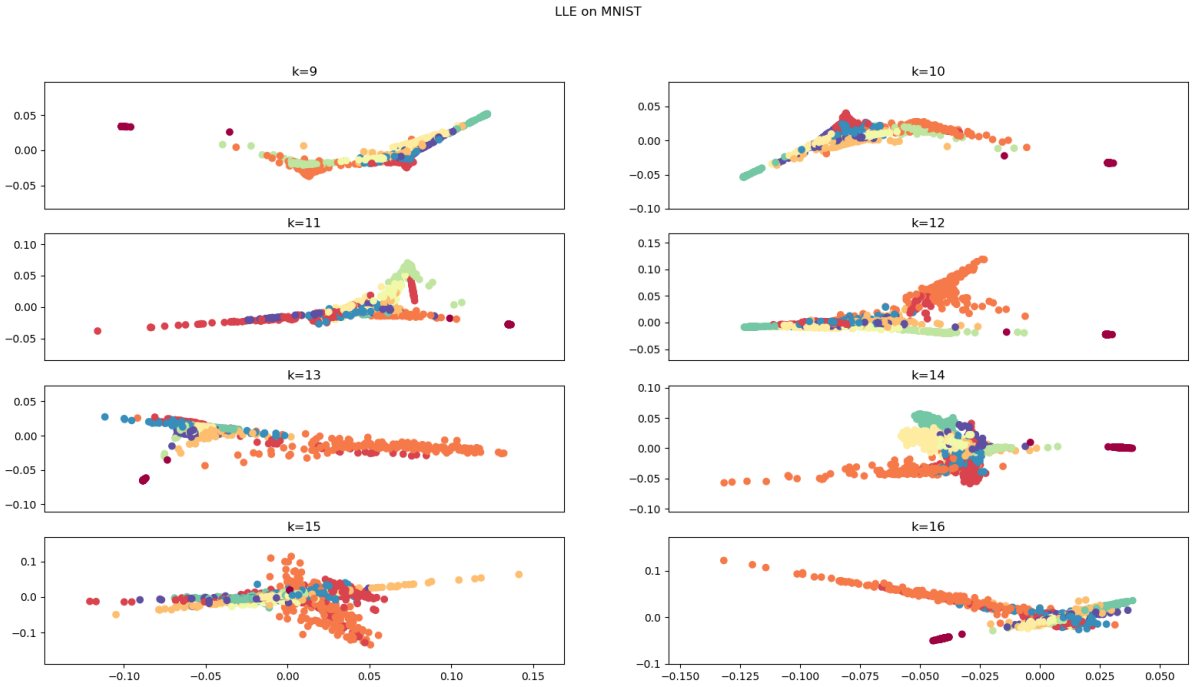

We can see that although the data lies on a 2-d subspace, PCA doesn't single out 2 eigenvalues (for any amount of noise) while MDS does this. Therefore we conclude that the eigenvalues in MDS are better scaled, and therefore we see better seperation. The eigenvalues in the PCA method measure the variance of the data for each direction (when direction is determined by the corresponding eigenvectors), while in MDS they measure the variance of the distances for each direction. Therefore we can see that MDS singles out 2 directions relevant to the distance (which is the case since the data lies on a 2-d space) while PCA only finds one direction which is relevant to the data. For both methods we see that adding noises smooths the eigenvalue curve, which was expected since the noise isn't necessarily contained in the subspace the original data is in.

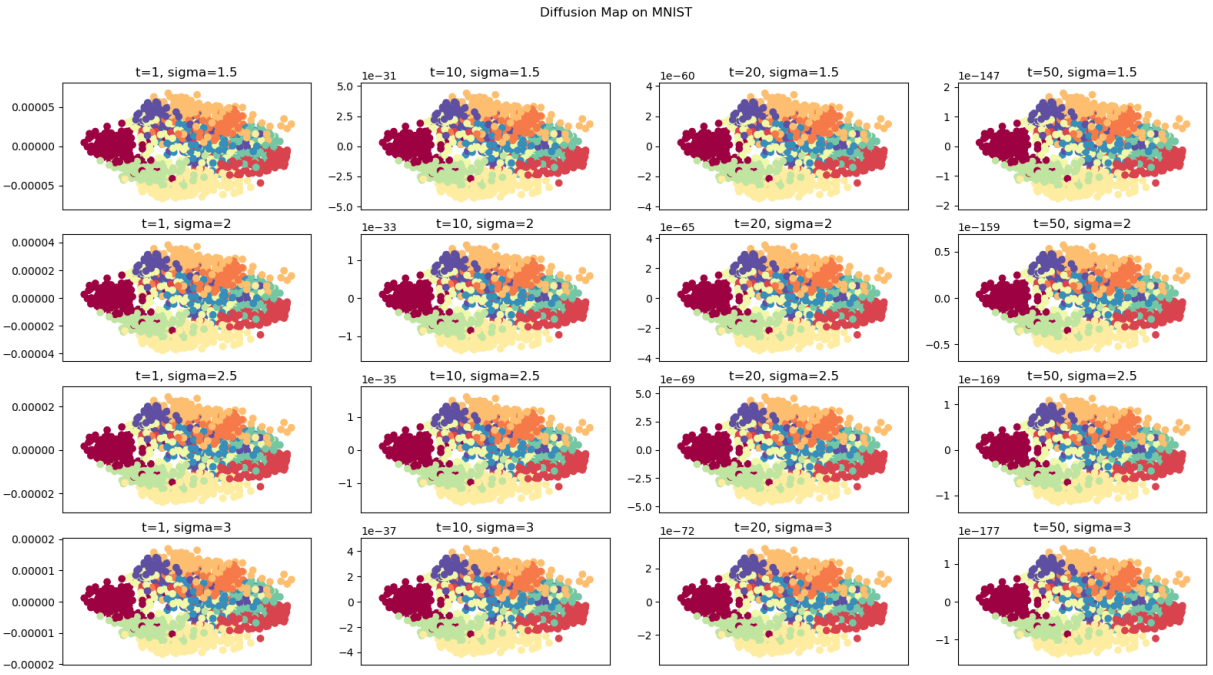
## MNIST

In the following plots we can see that MDS (and DM) perform alot better than LLE on the MNIST dataset (and all three perform worse than tSNE, as we saw in the last exercise). What we understand from this is that the euclidian distance between images is a more dominant parameter in determining the data's clustering (which point is an image of what number) than the linear locality of the data. For DM we can see that general clusters are formed, but are not seperated so good. This might be because the euclidian distances between different clusters of images aren't large, and are therefore preserved (and even squeezed

due to the effect of dimensionality recution) in the lower dimension.

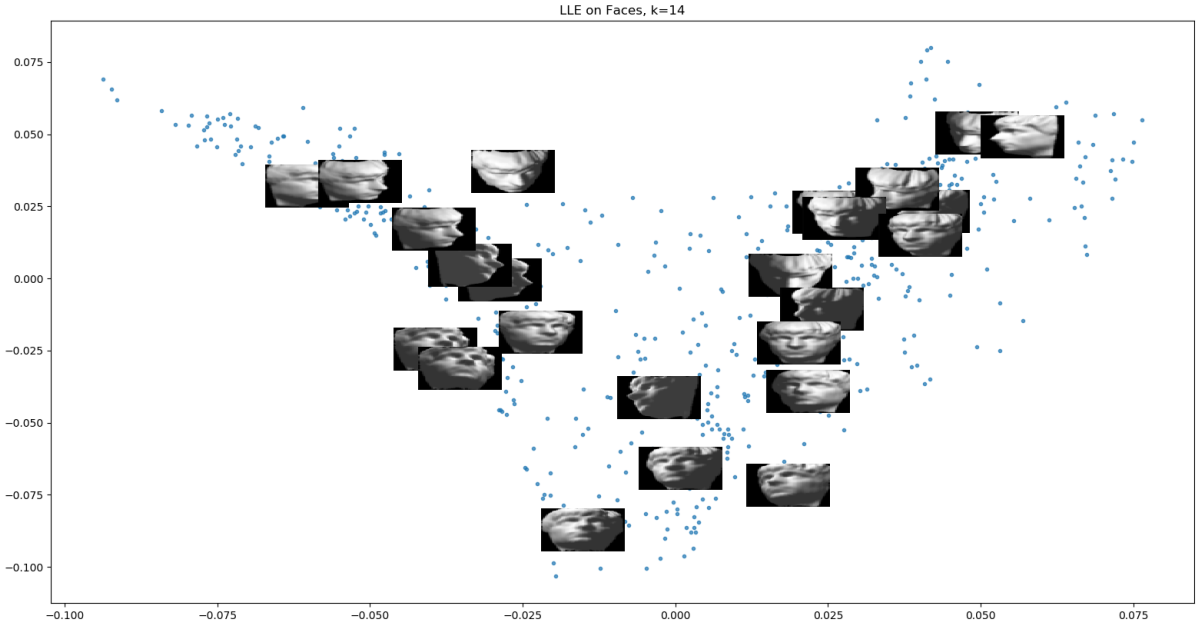



LLE on MNIST
k=9
k=10
k=11
k=12
k=13
k=14
k=15
k=16

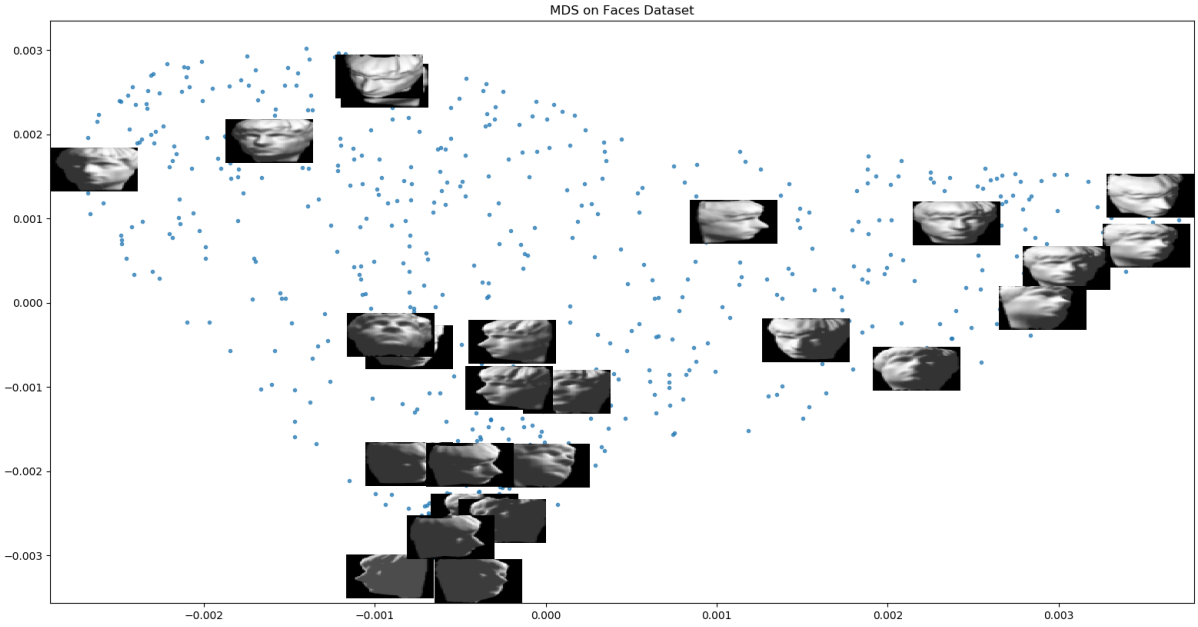



## Faces Dataset

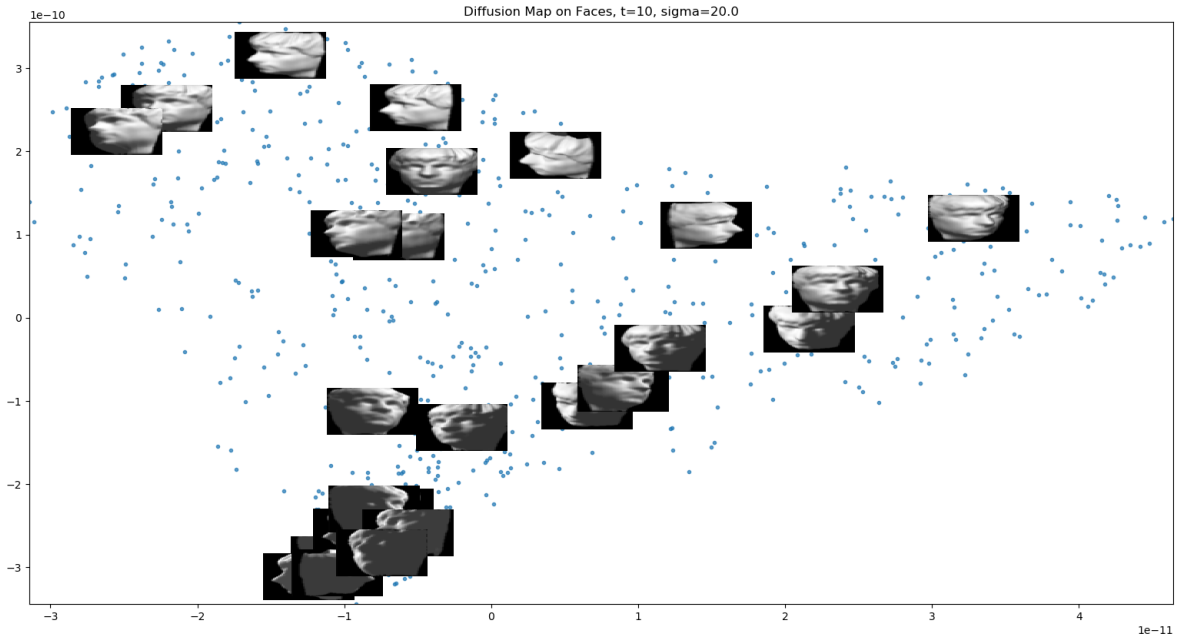
In the following figures (produced after parameter tweaking), we can see that all three methods manage to learn certain parameters of the data. Of course the faces presented are a relatively small sample of the data, but we can still see some trends.




we can see that LLE is able to roughly recognize the angle of the face. In the lower dimension this is roughly measured by the x axis - faces looking left are on the left, faces looking straight are in the middle and looking right are on the right.




We can see above that MDS learns a bit more about the lighting of the faces, where poorly lit images are clustered together, relative to better lit images.




We can see that DM learns both parameters mentioned above - both lighting and angle of the face. Again faces looking left are on the left and those looking right are on the right, and poorly lit images are also all clustered together.