DD2434 - Advanced Machine Learning - Assignment # 1

Answers to Theoretical Questions

&

Dimensionality Reduction with the UCI 'Zoo' Dataset

Altay Dikme

December 1, 2020

Contents

1 Problem 1

We let $\mathbf{A}$ be a real $n \times n$ symmetric matrix. $\mathbf{A}$ is *positive semidefinite* if and only if $\mathbf{x^T A x} \geq 0$ for every vector $\mathbf{x} \in \mathbb{R}^n$. We want to prove the following statements:

1. Prove that a real symmetric matrix has real eigenvalues.
2. Prove that a real symmetric matrix has orthogonal eigenvectors. Then prove that the eigen-decomposition of a real symmetric matrix is $\mathbf{A = ULU^T}$
3. Prove that a positive semidefinite matrix has non-negative eigenvalues.
4. Let $\mathbf{A}$ be a positive semidefinite matrix. Define matrix $\mathbf{D}$ so that $\mathbf{D}_{ij} = \mathbf{A}_{ii} + \mathbf{A}_{jj} - 2\mathbf{A}_{ij}$. Show that there exists n vectors $\mathbf{v}_1, \ldots, \mathbf{v}_n$ in $\mathbb{R}^n$ so that $\mathbf{D}_{ij} = ||\mathbf{v}_i - \mathbf{v}_j||^2$.

1.1

We begin with the first statement. If $\mathbf{A}$ is a real symmetric matrix it means that we have $\mathbf{A} = \mathbf{A}^T$. If we let λ be an eigenvalue of $\mathbf{A}$ and $\mathbf{v}$ the corresponding eigenvector we have:

$$\mathbf{Av} = \lambda \mathbf{v} \tag{1}$$

Multiplying both sides from the left with $\bar{\mathbf{v}}^T$:

$$\bar{\mathbf{v}}^T \mathbf{Av} = \lambda \bar{\mathbf{v}}^T \mathbf{v} = \lambda ||\mathbf{v}|| \tag{2}$$

Taking the complex conjugate transpose of both sides:

$$\bar{\mathbf{v}}^T \bar{\mathbf{A}}^T \mathbf{v} = \bar{\lambda} ||\mathbf{v}|| \tag{3}$$

Since we know that $\mathbf{A}$ i s real symmetric we have as earlier mentioned $\mathbf{A} = \mathbf{A}^T$ and $\mathbf{A} = \bar{\mathbf{A}}$. Thus we can re-write the left hand side as:

$$\bar{\mathbf{v}}^T \bar{\mathbf{A}}^T \mathbf{v} = \bar{\mathbf{v}}^T \mathbf{Av} \tag{4}$$

Thus we have:

$$\bar{\mathbf{v}}^T \mathbf{Av} = \bar{\lambda} ||\mathbf{v}|| \tag{5}$$

and (Eq. 3)

$$\bar{\mathbf{v}}^T \mathbf{Av} = \lambda ||\mathbf{v}|| \tag{6}$$

From this it follows that:

$$\bar{\lambda} ||\mathbf{v}|| = \lambda ||\mathbf{v}|| \tag{7}$$

Since $\mathbf{v}$ is an eigenvector, $||\mathbf{v}|| \neq 0$, and we can get rid of it leaving us with $\lambda = \bar{\lambda}$ which implies that λ is a real number. Thus our eigenvalues are real numbers. □

1.2

Next we want to show that a real symmetrix matrix has orthogonal eigenvectors, and prove that the eigen-decomposition of a real symmetric matrix is $\mathbf{A = ULU^T}$. We let λ and ζ be two distinct eigenvalues of $\mathbf{A}$ and $\mathbf{v}$ and $\mathbf{u}$ be their corresponding eigenvectors. We want to show that $\mathbf{v} \cdot \mathbf{u} = 0$, i.e that the eigenvectors are orthogonal. We begin with multiplying the dot product with the eigenvalue λ giving us:

$$\lambda(\mathbf{v} \cdot \mathbf{u}) = (\lambda \mathbf{v}) \cdot \mathbf{u} = \mathbf{Av} \cdot \mathbf{u} = (\mathbf{Av})^T \mathbf{u} = \mathbf{v}^T \mathbf{A}^T \mathbf{u} \tag{8}$$

Here we can use the fact that $\mathbf{A = A^T}$ giving us:

$$\rightarrow \mathbf{v^T A u} = \mathbf{v^T} \zeta \mathbf{u} = \zeta \mathbf{v^T u} = \zeta(\mathbf{v} \cdot \mathbf{u}) \tag{9}$$

I.e we have shown that $\lambda(\mathbf{v} \cdot \mathbf{u}) = \zeta(\mathbf{v} \cdot \mathbf{u})$ and since these are two distinct eigenvalues, we have $\lambda \neq \zeta$ which in turn means that we must have $\mathbf{v} \cdot \mathbf{u} = 0$. Thus showing that distinct eigenvalues of a real symmetric

matrix has orthogonal eigenvectors.

But what about degenerate eigenvalues, i.e eigenvalues with algebraic multiplicity greater than 1? In this case we still know that the eigenvectors of the degenerate eigenvalue will still be orthogonal to the *other* eigenvectors corresponding to other eigenvalues (Just proven), however we do not know if we will have "enough" eigenvectors to form a basis for decomposition later on. This depends on the geometric multiplicity of the eigenvalue, i.e the dimension of the space spanned by the eigenvectors of the degenerate eigenvalue. If the geometric multiplicity is equal to the algebraic multiplicity $m_g = m_a$ then it is possible to construct a space using the Gram-Schmidt method where the two eigenvectors of the degenerate eigenvalue are orthogonal. Thus it is always possible to *construct* an eigenspace in which all eigenvectors are orthogonal for a real symmetric matrix. In the case of having no degenerate eigenvalues, there is no need to construct such a space because all eigenvectors will be orthogonal "naturally" so to say.

Furthermore we want to prove that the eigen-decomposition of a real symmetric matrix $\mathbf{A} = \mathbf{ULU^T}$. During the Q&A session it was clarified that using a known proof is allowed, and thus I shall use the proof provided by Carleton university.[1]

Let the matrix $\mathbf{A}$ be an $n \times n$ matrix. The case where $n = 1$ is trivial. We suppose that for $n = k - 1$ that $\mathbf{A}$ is diagonalizable (or in other words has the eigen-decomposition $\mathbf{A} = \mathbf{QLQ^T}$). Using proof by induction we must now prove that this holds also for $n = k$. The characteristic polynomial of $\mathbf{A}$ ($k \times k$ real symmetric matrix) has at least one root λ_1 and thus at least one eigenvalue. We let $\mathbf{b}_1$ be an eigenvector of the eigenvalue λ_1, and rescale it to make it of unit length. We then pick vectors $\mathbf{b}_2, \ldots, \mathbf{b}_k$ so that $\mathbf{b}_1, \ldots, \mathbf{b}_k$ forms an orthonormal basis. (Can be done using Gram-Schmidt). We form $\mathbf{Q}$ where the columns are the $\mathbf{b}_i$'s. Then we have:

$$\mathbf{AQ} = [\lambda_1 \mathbf{b}_1, A\mathbf{b}_2, \ldots, A\mathbf{b}_k] = \mathbf{Q} \begin{bmatrix} \lambda_1 & d \\ 0 & A' \end{bmatrix} \tag{10}$$

for some $d \in \mathbb{R}^{1\times(k-1)}$ and $A' \in \mathbb{R}^{(k-1)\times(k-1)}$. Furthermore since $\mathbf{Q}$ is orthonormal (by construction) we have:

$$\mathbf{Q^T AQ} = \begin{bmatrix} \lambda_1 & d \\ 0 & A' \end{bmatrix} \tag{11}$$

Transposing this gives us:

$$\mathbf{Q^T A^T Q} = \begin{bmatrix} \lambda_1 & 0 \\ d^T & A'^T \end{bmatrix} \tag{12}$$

Since $\mathbf{A}$ is symmetric we have $\mathbf{A} = \mathbf{A^T}$, therefore $\mathbf{Q^T AQ} = \mathbf{Q^T A^T Q}$, therefore $d = 0$ and $A' = A'^T$. Also since A' is an $(k-1) \times (k-1)$ real symmetric matrix, by our induction hypothesis, there exists Q', L', Q'^T such that $A' = Q'L'T'^T$ (I.e eigendecomposable). Therefore we have:

$$\mathbf{Q^T AQ} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & Q'L'T'^T \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & Q' \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & L' \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & Q'^T \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & Q' \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & L' \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & Q' \end{bmatrix}^T \tag{13}$$

From this we can see that:

$$\mathbf{A} = \mathbf{Q} \begin{bmatrix} 1 & 0 \\ 0 & Q' \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & L' \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & Q' \end{bmatrix}^T \mathbf{Q^T} \tag{14}$$

We can now set:

$$\mathbf{U} = \mathbf{Q} \begin{bmatrix} 1 & 0 \\ 0 & Q' \end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & L' \end{bmatrix} \tag{15}$$

From this we see that we have $\mathbf{A} = \mathbf{ULU^T}$ which is what we wanted to show. The only thing remaining to be done is to show that $\mathbf{U^T U} = \mathbf{I}$. We have:

$$\mathbf{U^T U} = \begin{bmatrix} 1 & 0 \\ 0 & Q' \end{bmatrix}^T \mathbf{Q}^T \mathbf{Q} \begin{bmatrix} 1 & 0 \\ 0 & Q' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \mathbf{1}_{k-1} \end{bmatrix} = \mathbf{I}_k \tag{16}$$

□

[1] https://people.math.carleton.ca/~kcheung/math/notes/MATH1107/wk10/10_orthogonal_diagonalization_proof.html

1.3

We want to show that a positive semidefinite matrix has non-negative eigenvalues. If we let λ be an eigenvalue of $\mathbf{A}$ and $\mathbf{v}$ the corresponding eigenvector we have:

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v} \tag{17}$$

Multiplying both sides from the left with $\mathbf{v}^T$ gives us:

$$\mathbf{v}^T\mathbf{A}\mathbf{v} = \lambda\mathbf{v}^T\mathbf{v} = \lambda||\mathbf{v}||^2 \tag{18}$$

We know that both the left hand side and right hand side of this expression are positive. The left due to $\mathbf{A}$ being positive semidefinite and the right due to the eigenvector being a non-zero vector. Thus λ must also be real, and we have shown that a positive semidefinite matrix has non-negative eigenvalues. □

1.4

Finally we want to show that if $\mathbf{A}$ is a positive semidefinite matrix, and we define $\mathbf{D}_{ij} = A_{ii} + A_{jj} - 2A_{ij}$, and show that there exists n vectors $\{\mathbf{v_1}, ..., \mathbf{v_n}\}$ in $\mathbb{R}^n$ so that $\mathbf{D}_{ij} = ||\mathbf{v}_i - \mathbf{v}_j||^2$. Since we know that $\mathbf{A}$ can be written as an eigendecomposition this means that we have:

$$\mathbf{A} = \mathbf{QSQ^T} = (\mathbf{S}^{\frac{1}{2}}\mathbf{Q}^T)^T(\mathbf{S}^{\frac{1}{2}}\mathbf{Q}^T) \tag{19}$$

We can now let the vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ be the columns of $\mathbf{S}^{\frac{1}{2}}\mathbf{Q}^T$. We then have $\mathbf{A} = \mathbf{V^T V}$ where $\mathbf{V} = \mathbf{S}^{\frac{1}{2}}\mathbf{Q^T}$ We can now write A in index notation. We have:

$$A_{ii} = (V_{ik})^T V_{ki} = V_{ki}V_{ki} \tag{20}$$
$$A_{jj} = (V_{jk})^T V_{kj} = V_{kj}V_{kj} \tag{21}$$
$$A_{ij} = (V_{ik})^T V_{kj} = V_{ki}V_{kj} \tag{22}$$
$$A_{ji} = (V_{jk})^T V_{ki} = V_{kj}V_{ki} \tag{23}$$

Then we have:

$$\mathbf{D}_{ij} = A_{ii} + A_{jj} - 2A_{ij} = A_{ii} + A_{jj} - A_{ij} - A_{ji} \tag{24}$$
$$= V_{ki}V_{ki} + V_{kj}V_{kj} - V_{ki}V_{kj} - V_{kj}V_{ki} \tag{25}$$
$$= (V_{ki} - V_{kj})(V_{ki} - V_{kj}) = (V_{ik} - V_{jk})^T(V_{ki} - V_{kj}) = ||\mathbf{v}_i - \mathbf{v}_j||^2 \tag{26}$$

Which is what we wanted to show. □

2 Problem 2

We have a data matrix of dimension $m \times n$, where it may represent "document×words" and we may want to perform PCA both with respect to the rows and the columns. The question is if a *single* SVD operation is sufficient to perform PCA both on the rows and the columns of the data matrix. The reasoning is that for SVD we have: $(\mathbf{USV^T})^\mathbf{T} = \mathbf{VS^TU^T} = \mathbf{VS'U^T}$.

Inspecting the dimensions, we begin with the data matrix $\mathbf{Y}$ which has dimensions $m \times n$, thus after SVD we will have matrices U, S, V^T with dimensions $m \times m$, $m \times n$, and $n \times n$ respectively. By the argument given in the question the transpose of the SVD will have dimensions $n \times n$, $n \times m$ and $m \times m$ respectively ($\mathbf{VS'U^T}$).

So now we want to see if these dimensions will be the same as if we perform SVD on the transpose of our data matrix. In this case our data matrix will have dimensions $n \times m$ and the singular value decomposition should have dimensions $n \times n$, $n \times m$ and $m \times m$, which are the dimensions we get when transposing the SVD of the data matrix in the original case. Thus we may think that the argument given in the question does imply that a single SVD operation is sufficient to perform PCA on both the rows and the columns of a

data matrix, however this is not true since we are forgetting about a step used in PCA.

In general we want to center our data points before we perform the SVD and thus if we assume that centering has been performed on our rows, it does not necessarily mean that our *columns* are centered. Due to this simple reason a single SVD operation is not sufficient to perform PCA on both the rows and the columns of the data matrix. The exception of course is if both the rows and columns are centered, in which case we are perfectly fine.

3 Problem 3

The first claim is that the maximizer of the expression $\text{tr}\big(\mathbf{Y^T W W^T Y}\big)$ is given by $\mathbf{W} = \mathbf{U}_k$, i.e by the k left singular vectors of $\mathbf{Y}$ associated with the k largest singular values. We note that $\text{tr}\big(\mathbf{Y^T W W^T Y}\big) = \mathbb{E}_y[||\mathbf{W^T Y}||^2]$, which is what we want to maximize. In other words we want to maximize $||\mathbf{W^T Y}||^2$ subject to $||\mathbf{W}||^2 \leq 1$. We can now use Lagrange multipliers and construct the Lagrangian:

$$L(\mathbf{W}, \lambda) = ||\mathbf{W^T Y}||^2 + \lambda(||\mathbf{W}||^2 - 1) \tag{27}$$

We want to extremize this Lagrangian, thus we can take the gradient with respect to $\mathbf{W}$ and setting the expression equal to zero giving us:

$$\nabla_{\mathbf{W}} L(\mathbf{W}, \lambda) = 2\mathbf{Y}\mathbf{Y^T}\mathbf{W} - 2\lambda\mathbf{W} = 0 \tag{28}$$

$$\rightarrow \mathbf{Y}\mathbf{Y^T}\mathbf{W} = \lambda\mathbf{W} \tag{29}$$

We see that this is simply an eigenvalue problem. The trace is maximized when $\mathbf{W}$ is the eigenvectors of $\mathbf{Y}\mathbf{Y^T}$. Furthermore we also know that the SVD of $\mathbf{Y}$ is $\mathbf{Y} = \mathbf{Q}\mathbf{S}\mathbf{V^T}$. where $\mathbf{Q}$ are the left singular vectors, $\mathbf{S}$ is a diagonal matrix consisting of the singular values in a descending order, and V^T are the right singular vectors. Inserting this we have:

$$\mathbf{Y}\mathbf{Y^T} = \mathbf{Q}\mathbf{S}\mathbf{V^T}\mathbf{V}\mathbf{S^T}\mathbf{Q^T} = \mathbf{Q}\mathbf{S}\mathbf{S^T}\mathbf{Q^T} \tag{30}$$

We also note that $\mathbf{D} = \mathbf{S}\mathbf{S^T}$ is just the diagonal matrix containing the eigenvalues of $\mathbf{Y}\mathbf{Y^T}$ in a descending order. Therefore we see that we can write $\mathbf{Y}\mathbf{Y^T} = \mathbf{Q}\mathbf{D}\mathbf{Q^T}$. We can thus see that the eigenvectors of $\mathbf{Y}\mathbf{Y^T}$ are the left singular vectors of the Singular Value decomposition of $\mathbf{Y}$.

As seen in Eq. 29 we see that the trace is maximized when $\mathbf{W}$ is given by the eigenvectors of $\mathbf{Y}\mathbf{Y^T}$, which was just shown to be the left singular vectors of $\mathbf{Y}$, i.e $\mathbf{W} = \mathbf{U}$, or for the use in PCA we want to select k number of principal components, thus we have $\mathbf{W} = \mathbf{U}_k$ where $\mathbf{U}_k$ are the k left singular vectors of $\mathbf{Y}$ associated with the k largest singular values. Thus we have shown the first claim made in the assignment. □

The second claim made in the assignment was that the maximum value achieved is $\sum_{i=1}^{k} \sigma_i^2$, where σ_i are assumed to be the singular values of $\mathbf{Y}$. This follows easily from our previous claim. We showed that the trace was maximized when $\mathbf{W} = \mathbf{U}_k$, and thus we can just plug this into the trace, and inserting the singular value decomposition of $\mathbf{Y}$ we have:

$$\text{tr}\big(\mathbf{V}\mathbf{S^T}\mathbf{U^T}\mathbf{U}\mathbf{U^T}\mathbf{U}\mathbf{S}\mathbf{V^T}\big) = \text{tr}\big(\mathbf{V}\mathbf{S^T}\mathbf{S}\mathbf{V^T}\big) = \text{tr}\big(\mathbf{V^T}\mathbf{V}\mathbf{S^T}\mathbf{S}\big) = \text{tr}\big(\mathbf{S^T}\mathbf{S}\big) = \sum_{i=1}^{k} \sigma_i^2 \tag{31}$$

Which is what we wanted to show. □

4 Problem 4

In the derivation of classical MDS we derived the Gram matrix from the distance matrix resulting in $s_{ij} = -\frac{1}{2}(d_{ij}^2 - s_{ii} - s_{jj})$. Now we claim that s_{ij} can also be computed as $s_{ij} = -\frac{1}{2}(d_{ij}^2 - d_{1i}^2 - d_{1j}^2)$ where d_{1i} and d_{1j} are the distances from the first point in the dataset to points i and j resp. We now want to argue

that the claim is correct, that is that it provides the correct estimate for the Gram matrix. This means that we want to be able to show that $\mathbf{S} = \mathbf{Y}^\mathbf{T}\mathbf{Y}$ and that $\mathbf{Y}$ gives 0 reconstruction error for $\mathbf{D}$. The claim in the question is valid due to the fact that we are dealing with *pairwise* distances, which has a translational invariance meaning that if you move all of the data it will inherently be unchanged. We begin by using the classic "trick" of adding and subtracting a constant (vector in this case).

$$d_{ij}^2 = ||\mathbf{y}_i - \mathbf{y}_j||^2 = ||\mathbf{y}_i - \mathbf{y}_j + \mathbf{y}_1 - \mathbf{y}_1||^2 = ||(\mathbf{y}_i - \mathbf{y}_1) + (\mathbf{y}_1 - \mathbf{y}_j)||^2 \tag{32}$$

$$= [(\mathbf{y}_i - \mathbf{y}_1) + (\mathbf{y}_1 - \mathbf{y}_j)]^T[(\mathbf{y}_i - \mathbf{y}_1) + (\mathbf{y}_1 - \mathbf{y}_j)] \tag{33}$$

$$= (\mathbf{y}_i - \mathbf{y}_1)^T(\mathbf{y}_i - \mathbf{y}_1) + (\mathbf{y}_i - \mathbf{y}_1)^T(\mathbf{y}_1 - \mathbf{y}_j) + (\mathbf{y}_1 - \mathbf{y}_j)^T(\mathbf{y}_i - \mathbf{y}_1) + (\mathbf{y}_1 - \mathbf{y}_j)^T(\mathbf{y}_1 - \mathbf{y}_j) \tag{34}$$

Can re-write the last term on the last row as $(\mathbf{y}_1 - \mathbf{y}_j)^T(\mathbf{y}_1 - \mathbf{y}_j) = (\mathbf{y}_j - \mathbf{y}_1)^T(\mathbf{y}_j - \mathbf{y}_1)$, and then write out in terms of the norm again:

$$d_{ij}^2 = ||\mathbf{y}_i - \mathbf{y}_1||^2 + ||\mathbf{y}_j - \mathbf{y}_1||^2 + (\mathbf{y}_i - \mathbf{y}_1)^T(\mathbf{y}_1 - \mathbf{y}_j) + (\mathbf{y}_1 - \mathbf{y}_j)^T(\mathbf{y}_i - \mathbf{y}_1) \tag{35}$$

$$= d_{1i}^2 + d_{1j}^2 + (\mathbf{y}_i - \mathbf{y}_1)^T(\mathbf{y}_1 - \mathbf{y}_j) + (\mathbf{y}_1 - \mathbf{y}_j)^T(\mathbf{y}_i - \mathbf{y}_1) \tag{36}$$

Now we can use the translational invariance which was stated earlier. Since it does not matter *where* our data is (can be shifted anywhere) we can choose to center our data around $\mathbf{y}_1$. This then means that we can set $\mathbf{y}_1 = 0$ giving us:

$$d_{ij}^2 = d_{1i}^2 + d_{1j}^2 - \mathbf{y}_i^T\mathbf{y}_j - \mathbf{y}_j^T\mathbf{y}_i = d_{1i} + d_{1j} - s_{ij} - s_{ji} \tag{37}$$

Furthermore we know that $s_{ji} = s_{ij}$ (Symmetric), we have:

$$d_{ij}^2 = d_{1i}^2 + d_{1j}^2 - 2s_{ij} \Rightarrow s_{ij} = -\frac{1}{2}(d_{ij}^2 - d_{1i}^2 - d_{1j}^2) \tag{38}$$

Which is what we wanted to show. Since we constructed s_{ij} from $\mathbf{Y}$ we know that Eq. 38 will fullfill $\mathbf{S} = \mathbf{Y}^\mathbf{T}\mathbf{Y}$ and that we will have 0 reconstruction error. □

5 Problem 5

In the classical MDS algorithm when $\mathbf{Y}$ is known we form $\mathbf{S} = \mathbf{Y}^\mathbf{T}\mathbf{Y}$ and obtain the MDS embedding bby the eigenvalue decomposition of $\mathbf{S}$. We want to show that this procedure is equivalent to performing PCA on $\mathbf{Y}$.

The reason why these are equivalent is due to the relationship between singular value decomposition and eigenvalue decomposition. When we perform PCA on $\mathbf{Y}$ we have the SVD $\mathbf{Y} = \mathbf{U}\mathbf{S}\mathbf{V}^\mathbf{T}$ where if we let $\mathbf{Y}$ be an $m \times n$ matrix, then our decomposed matrices will as stated for previous questions have dimensions $m \times m$, $m \times n$ and $n \times n$ respectively. Now if we look at $\mathbf{Y}^\mathbf{T}\mathbf{Y}$ written in terms of the SVD of $\mathbf{Y}$ we have:

$$\mathbf{Y}^\mathbf{T}\mathbf{Y} = \mathbf{V}\mathbf{S}^\mathbf{T}\mathbf{U}^\mathbf{T}\mathbf{U}\mathbf{S}\mathbf{V}^\mathbf{T} = \mathbf{V}(\mathbf{S}^\mathbf{T}\mathbf{S})\mathbf{V}^\mathbf{T} = \mathbf{V}(\mathbf{S}^\mathbf{T}\mathbf{S})\mathbf{V}^{-\mathbf{1}} \tag{39}$$

We can see that this is the eigenvalue decomposition of $\mathbf{S} = \mathbf{Y}^\mathbf{T}\mathbf{Y}$, where the columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{Y}^\mathbf{T}\mathbf{Y}$ and $(\mathbf{S}^\mathbf{T}\mathbf{S})$ is the diagonal matrix with eigenvalues of $\mathbf{Y}^\mathbf{T}\mathbf{Y}$. Thus if we perform PCA on $\mathbf{Y}$ we get the same information as performing the eigenvalue decomposition on $\mathbf{S} = \mathbf{Y}^\mathbf{T}\mathbf{Y}$.

The question now is which way is the best way to perform the embedding computationally? When performing PCA on $\mathbf{Y}$ we simply perform SVD on $\mathbf{Y}$, whereas when performing MDS on $\mathbf{S}$ we first have to multiply $\mathbf{Y}^\mathbf{T}\mathbf{Y}$ and then perform an eigendecomposition. For an $n \times n$ matrix the singular value decomposition has a computational complexity of $O(n^3)$. For MDS we first have a matrix multiplication which has computational complexity $O(n^3)$ (If we take a naive approach). Furthermore eigendecomposition (in Matlab) has computational complexity of $O(n^3)$ as well.

Thus we can see that for PCA we have a complexity of $O(n^3)$ and for MDS we have $O(2n^3)$, and it is "cheaper" to perform PCA, however in orders of magnitude they are approximately the same. Another reason why one would prefer to perform PCA on $\mathbf{Y}$ rather than MDS on $\mathbf{Y^T Y}$ is also due to numerical precision. When performing the matrix multiplication $\mathbf{Y^T Y}$ one may only keep up to a certain number of significant figures in the resulting matrix. Thus it may be better to just perform a PCA on $\mathbf{Y}$ instead.

Note that when answering the complexity part of this question we did not take into account the complexity of any centering of the data, and just looked at SVD vs matrix multiplication and eigendecomposition.

6 Problem 6

It is easy to see how the process to obtain the neighborhood graph G in the Isomap method may yield disconnected graphs. An example is shown in the figure below, where each point is connected to its *two* nearest neighbors. This is obviously not good since the Isomap method does not work when you have disconnected neighborhood graphs.

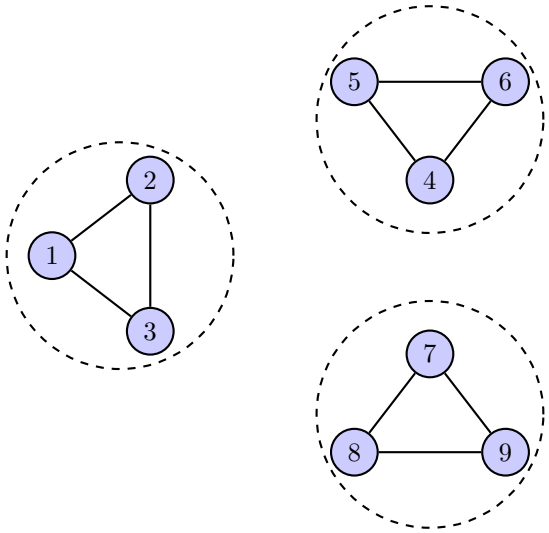


Figure 1: An example of how graphs can be disconnected. We have 2-nearest-neighbor coupling of the points. We see how this forms two separate graphs.

One way to patch this problem is to look at the disconnected graphs and note that since these data points are close to each other, we can infer that they are similar in some sense. One could approximate these points to be one single point. One way of doing this is to find the geometric center of the disconnected graph and create a new pseudo data point there which represents all n number of points in the graph. Applying this method on the three disconnected graphs in Fig. 1 will then result in Fig. 2.
We see in Fig. 2 that since the disconnected graphs have become new "points", all three "cluster" points are connected to each other, and thus we have patched the problem. In order to implement this patch in practice one would first have to identify disconnected graphs, followed by the creation of a new point which represents the disconnected graph, and finally re-run the k-nearest-neighbors algorithm where the points constituting the disconnected graph are replaced by the new cluster point.

So what are some drawbacks of using this patch method? One problem arises from our assumption that points in near proximity to each other. If the size of the disconnected graph is large, i.e the points of the disconnected graph are spread out then clustering all of these points into a single point will result in loss of information.

Another possible fix to the problem of disconnected graphs is to artificially connect the points of the disconnected graphs to each other. This would be done by iterating through each point of each disconnected

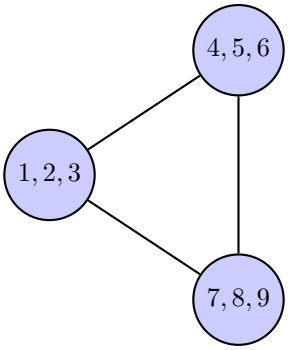


Figure 2: The three disconnected graphs all become connected after applying the proposed method.

graph and computing the pairwise distances, and finally adding a connection from the two points which have the smallest pairwise distance. By doing this however we will have a final neighborhood graph which is not strictly k-Nearest-Neighbors since some points will have an extra neighbor which it is connected to. If this method were to be applied to the example in Fig. 1 it would lead to the connected graph shown in Fig. 3, and as we can see points 2,5, and 4 are connected to 3 other points and not 2 (Initially we had 2-Nearest-Neighbors).

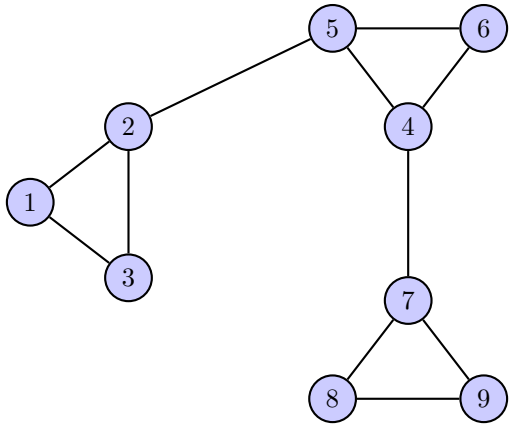


Figure 3: Alternative 'fix' to our problem of disconnected graphs. Will lead to neighborhood graphs which are not strictly k-nearest -neighbors.

7 Problem 7

https://github.com/AltayDikme-R/DD2434-Advanced-Machine-Learning

This is the progarmming task of the homework assignment, where we use the "zoo" dataset given in the UCI ML repository. The aim is to project the data in 2D so that "similar" animals are projected near each other. We are asked to do this using the three different methods of dimensionality reduction which we have learned so far; PCA, MCD and Isomap. Firstly it is important to note that the data consists of 17 (Excluding animal names) dimensions/categories of which all but one have boolean values. The category without booleans is the "legs" category which has the number of legs each animal has. So before we do anything we have to decide what we want to do with this category in order to make sure that each category only has boolean values.

The category includes the leg numbers of $0, 2, 4, 6, 8$, so one solution is to remove it and add five new categories, where each one corresponds to the number of legs. Thus if an animal has 0 legs it will have a boolean value of 1 for that category, or if it has 2 legs it will have boolean 1 and so on. Another solution is to change the category from "number of legs" to "has legs" so that if an animal has *any* number of legs it will have a value of 1 and 0 otherwise. Both of these methods will be implemented.

The last category "type" is removed from the dataset, per the assignment instructions. It will be used later as a color for each point in our visualization. Thus the data we will work with is either 20×101 or 16×101 depending on which way we decide to deal with the "legs" category. We denote these Dataset 1 and 2 respectively for clarity in the report.

7.1 Principal Component Analysis (PCA)

We begin by importing the dataset from the given repository, and note that the form of the data is such that each *row* is a datapoint (101×17). We would like each column to be a datapoint since that will bring the matrices to a form which we recognize from lectures and make visualizing the data easier (for me). This is easily done by transposing the data. Thus we have our data as a 17×101 matrix (101 different animals). The first step is to remove the last category (last row) as explained in the problem instructions. Next we implement the first solution as explained above for the "legs" category, where we create five new categories (rows). We also implement the other strategy. Thus we are working with two sets of data in parallel, one which is of dimensions 20×101 (Dataset 1) and the other is 16×101 (Dataset 2). Next we wish to center our data which in matrix notation is (given in the lectures):

$$\mathbf{Y} \leftarrow \mathbf{Y} - \frac{1}{n}\mathbf{Y}\mathbf{1}_n\mathbf{1}^{\mathbf{T}}{}_n \tag{40}$$

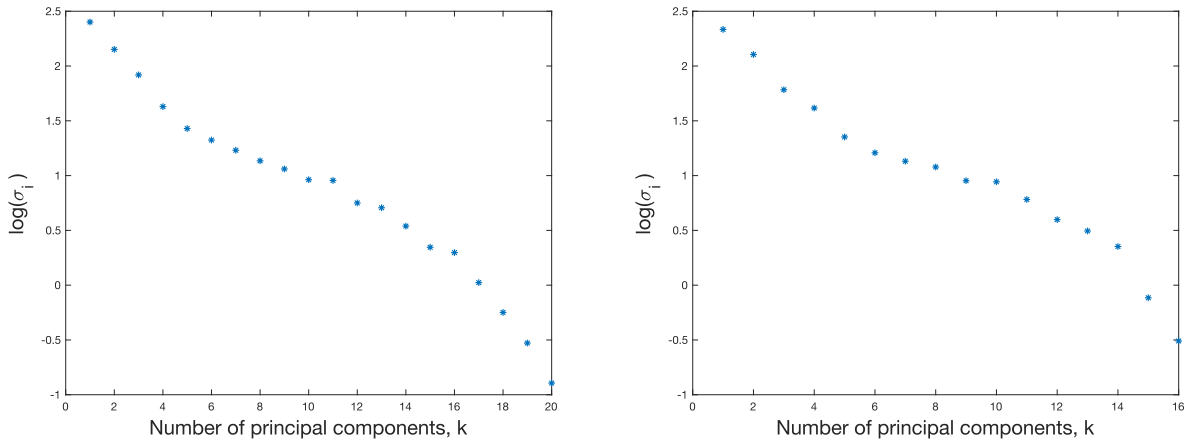
After centering the data we simply perform SVD on both of our data matrices. Since we wish to project the data in a two-dimensional plot we are going to choose two principal components. We also wanted to see how much of the data variance is explained when setting $k = 2$. For Dataset 1 we have:

$$\frac{\sum_{i=1}^{2}\sigma_i^2}{\sum_{i=1}^{20}\sigma_i^2} = 0.5429 \tag{41}$$

For Dataset 2 we have:

$$\frac{\sum_{i=1}^{2}\sigma_i^2}{\sum_{i=1}^{15}\sigma_i^2} = 0.5676 \tag{42}$$

We see that choosing $k = 2$ principal components explains 54.29% and 56.76% of the variance respectively. This falls well short of the "rule of thumb" given in the lecture where one wants to select k such that at least 85% of the variance is explained. In Fig. 4 we can see the logarithm of the singular values. From the plot one can deduce for both cases that perhaps choosing $k = 6$ would be a better choice, however it would not be practical in this case since we want a two-dimensional plot.



(a) Dataset 1

(b) Dataset 2

Figure 4: Logarithmic plot of singular values of both datasets.

Finally we plot the first two principal components and giving the points colors depending on the category "type" which was removed at the start. The results are shown in Figures 5 and 6. We can see that the principal components analysis does an all right job at projecting the data to a two-dimensional plane. It is assumed that the animals of a certain "type" are similar, and we can see that there is a large cluster of red points which is the majority of the dataset (Type 1), and some other clusters of the other types.

However we do see that some types such as type 3 (black points), are not clustered together, and that we have overlap between different types of animals (Type 2 and Type 6, Yellow and Purple). This happens for both Dataset 1 and Dataset 2. In general we would say that there is not a large difference in the classification power of Dataset 1 and 2. Both result in fairly similar plots, therefore one may deduce that creating extra dimensions for the number of legs may be superfluous.

7.2 Multidimensional Scaling (MDS)

For the multidimensional scaling (MDS) approach we once again begin with Dataset 1 and Dataset 2 which we "prepare" as explained in Section 7. We begin by creating a distance matrix by calculating the pairwise distances between the elements in our datasets. We are left with two distance matrices of size 101×101. From the distance matrix we wish to determine the simularity matrix $\mathbf{S}$, which can be done using the expression given in the lecture notes. Furthermore we want to use the double centering trick in order to make sure our data is centered in both the rows and columns. Thus we have our similarity matrix given by:

$$\mathbf{S} = -\frac{1}{2}\left(\mathbf{D} - \frac{1}{n}\mathbf{D}\mathbf{1}_n\mathbf{1}^{\mathbf{T}}{}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}^{\mathbf{T}}{}_n\mathbf{D} + \frac{1}{n^2}\mathbf{1}_n\mathbf{1}^{\mathbf{T}}{}_n\mathbf{D}\mathbf{1}_n\mathbf{1}^{\mathbf{T}}{}_n\right) \tag{43}$$

Next we perform an eigendecomposition of our similarity matrix $\mathbf{S}$ giving us our eigenvectors and eigenvalues. Finally we can find a k dimensional representation of the data by writing $\mathbf{X} = \mathbf{1}_{k\times n}\mathbf{S}^{\frac{1}{2}}\mathbf{U}^{\mathbf{T}}$, where $\mathbf{S}$ and $\mathbf{U}$ are the eigenvalues resp. eigenvectors. We once again want a two-dimensional representation therefore $k = 2$. In Figures 7 and 8 we can see the results of our multidimensional scaling. We see that the results are similar to the PCA results, where we can draw the same conclusions about the "legs" category. We also note that it clusters the same "similar" animals well and the same overlaps between types (e.g. Type 3 and 5). Thus the two methods perform similarly.

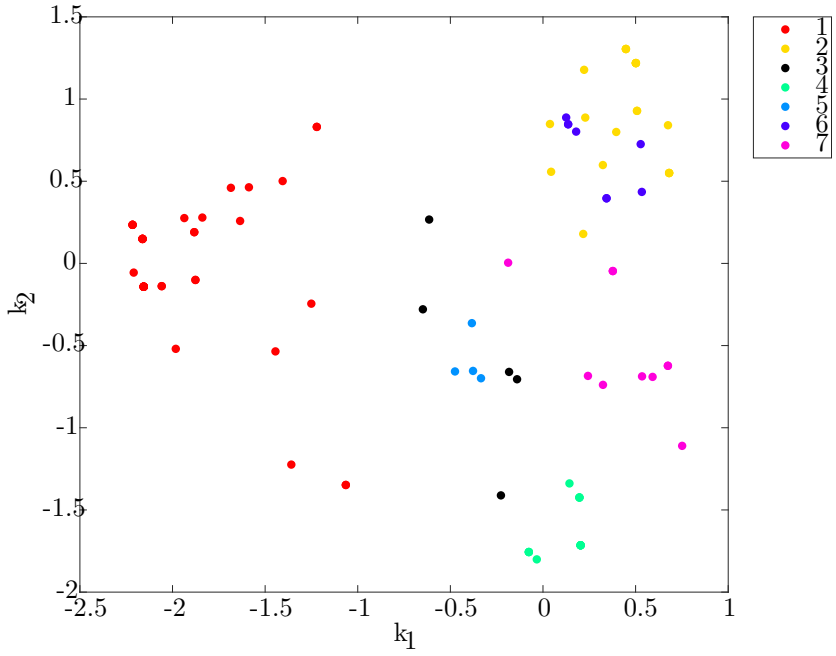


Figure 5: PCA - Dataset 1

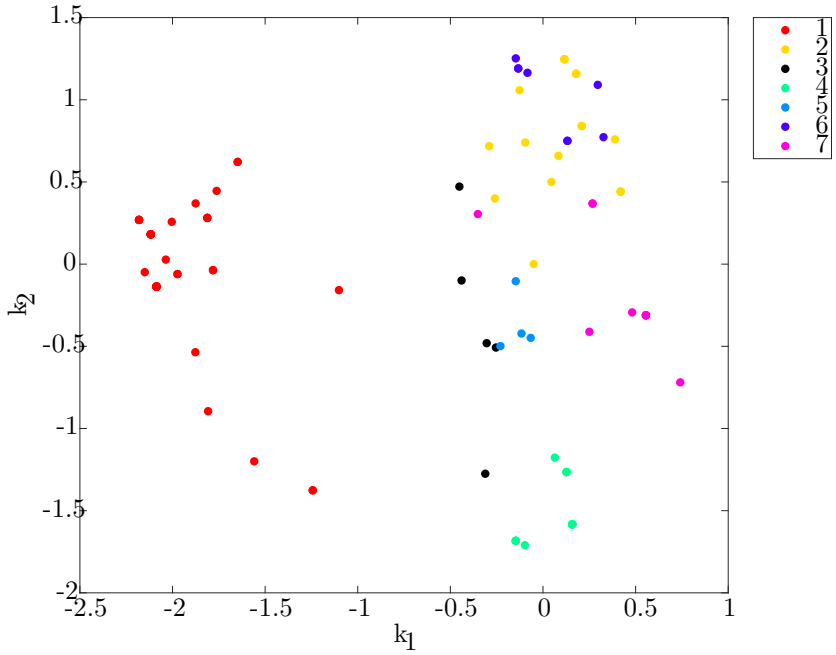


Figure 6: PCA - Dataset 2

7.3 Isomap

Finally for the Isomap approach we still begin with our two prepared datasets (Dataset 1, and Dataset 2 as explained in Section 7). As in the MDS case we begin by creating a distance matrix by calculating the

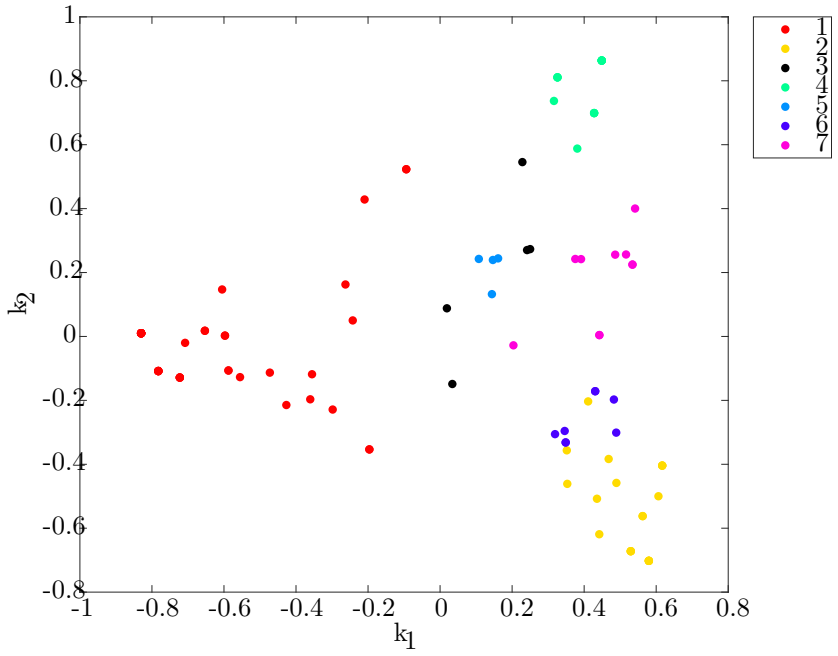


Figure 7: MDS - Dataset 1

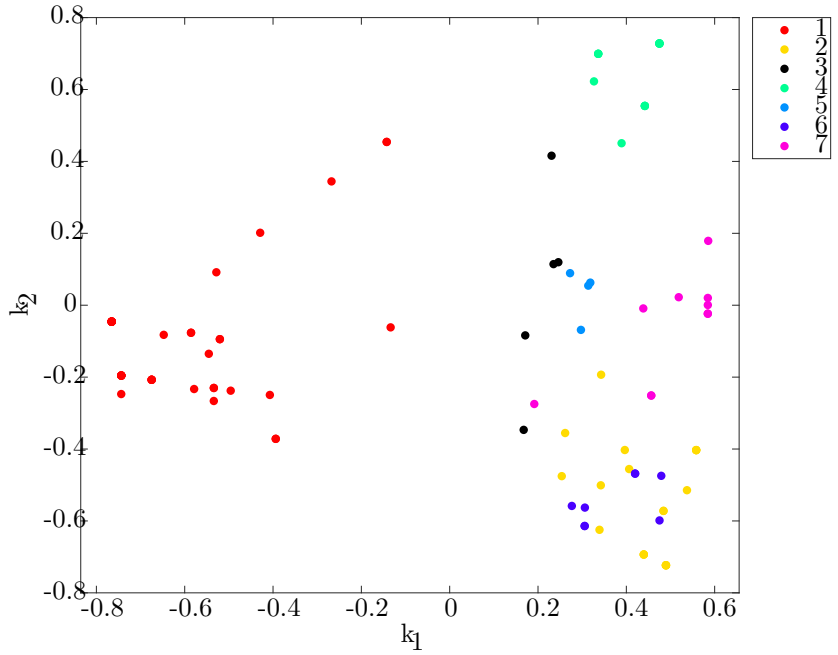


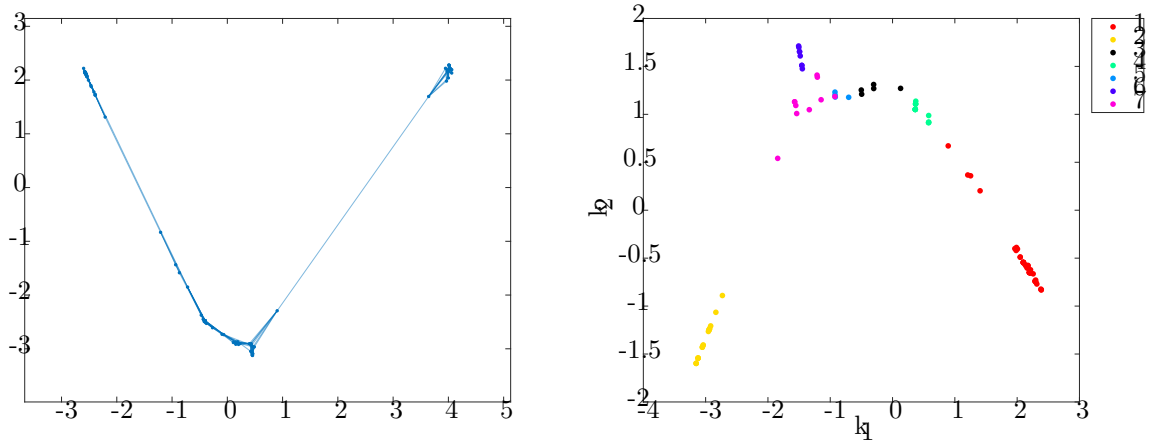
Figure 8: MDS - Dataset 2

pairwise distances between the elements in each respective dataset. One difference however is that this time we use the squared Euclidean distance. Next we wish to construct a neighborhood graph where each point is connected to its k nearest neighbors. At this step it is important to ensure that k is large enough so that

our neighborhood graph does not become disconnected, which we do by trial and error (We chose to not try and implement the heuristics given in Question 6 but rather just increase k until we found connected graphs). Once we have our neighborhood graph G, Djikstra's algorithm is used to determine the shortest path distances d_{ij} from points i to j. This results in a square symmetric matrix $\mathbf{D}$(Distance matrix). From here we can apply multidimensional scaling in a same manner as in the previous section using our new Distance matrix $\mathbf{D}$ as input.

7.3.1 Results Dataset 1

At first a small value for k was used, such as $k = 5$, however this resulted an some funky results. Turns out that this is due to the inherent problem with Isomap; that our neighborhood graphs were disconnected. By trial and error the first fully connected neighborhood graph was found for $k = 11$ (See Fig. 9a). After determining the shortest path distance using Djikstra's algorithm we performed MDS on the resulting distance matrix. The results are shown in Fig. 9b.

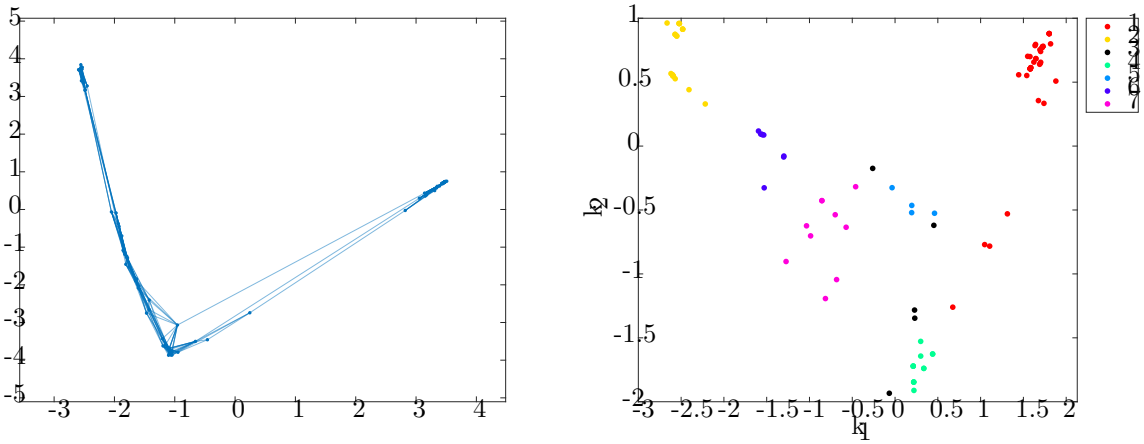


(a) Connected Neighborhood Graph for $k = 11$

(b) MDS results for $k = 11$

Figure 9: Neighborhood graph and MDS results for isomap implementation with $k = 11$.

We further increased the number of nearest neighbors to $k = 15$. The resulting neighborhood graph and MDS results are shown in Fig. 10. For $k = 15$ we observe that the MDS results seem to be better as we observe more seperated clusters of each color (type). The results appear to be the "best" of all methods which we have attempted so far.



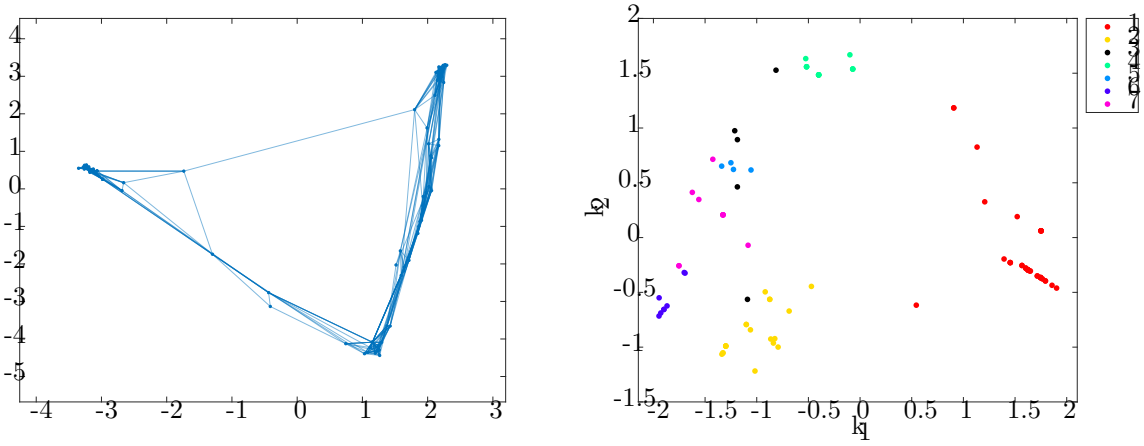
(a) Connected Neighborhood Graph for $k = 15$

(b) MDS results for $k = 15$

Figure 10: Neighborhood graph and MDS results for isomap implementation with $k = 15$. (Dataset 1)

7.3.2 Results Dataset 2

The same procedure as earlier was repeated for Dataset 2. In this case we note that the first connected graph occurs at $k = 9$. This may be due to the fact that this dataset only has 15 categories whereas Dataset 1 has 20. In order to compare our results with Dataset 1, we also ran the isomap implementation for $k = 15$ so that we can compare some "good" results. The resulting neighborhood graph and MDS results are shown in Fig. 11.



(a) Connected Neighborhood Graph for $k = 15$

(b) MDS results for $k = 15$

Figure 11: Neighborhood graph and MDS results for isomap implementation with $k = 15$. (Dataset 2)

We note that the MDS results for this dataset does not form as clear clusters as we had for Dataset 1. This may be an indication that separating the number of legs into separate categories may have an effect on our dimensionality reduction and thus our classification. This is in contrast with our previous methods of PCA and MDS where the difference between datasets was not too noticable.