Assignment 1

Methods of PCA, MDS and Isomap

DD2434 Advanced Machine Learning
Filip Bergentoft, bergento@kth.se

Problem 1

It is assumed throughout this problem that whenever a matrix A is referenced, it is a real and symmetric matrix of size $n \times n$.

Part (i): *Prove that a real symmetric matrix has real eigenvalues*

Let λ be an eigenvalue to A with a corresponding vector v, which is by the definition of an eigenvector non-zero.

The norm of a complex vector z is given by $\|z\| = \sqrt{z^T \overline{z}}$ and is non-zero for all $z \in \mathbb{C}$ except $z = 0$.

Given that A is real we can use that

$$\overline{Av} = \overline{\lambda v} = A\overline{v} = \overline{\lambda v} \tag{1}$$

Then we can expand $v^T A \overline{v}$ in the two following ways

$$v^T A\overline{v} = v^T(A\overline{v}) = \left\{ \text{Using equation (1)} \right\} = v^T \overline{\lambda v} = \overline{\lambda} v^T \overline{v} = \overline{\lambda}\|v\|^2 \tag{2}$$

$$v^T A\overline{v} = (A^T v)^T \overline{v} = \left\{ A = A^T \right\} = (Av)^T \overline{v} = \lambda v^T \overline{v} = \lambda \|v\|^2 \tag{3}$$

Thus since equation (2) and (3) are equal (from the left hand side) we get that

$$\overline{\lambda}\|v\|^2 = \lambda\|v\|^2 \Rightarrow \lambda = \overline{\lambda} \tag{4}$$

since v is non-zero by the definition of an eigenvector. Thus are the eigenvalues of a real and symmetric matrix real.

Part (ii): *Prove that a real symmetric matrix has orthogonal eigenvectors. Then, prove that the eigen-decomposition of a real symmetric matrix A is $A = Q\Lambda Q^T$*

Let $A \in \mathbb{R}^{⋉\times⋉}$ be a symmetric matrix. I will show this by using a proof of induction. The definition of an eigenvector v with an eigenvalue λ is that they fulfill the characteristic equation

$$(A - \lambda I)v = 0, \ v \text{ is non-trivial}$$

which implies that the kernel of $(A - \lambda I)$ is non-zero which in turn implies that

$$det(A - \lambda I) = 0$$

which is a polynomial equation of degree n with at least one root being $\lambda \in \mathbb{R}$ (because of the symmetric properties of A). Thus is $Av = \lambda v$ and we have at least one eigenvector v which is trivially orthogonal and the base case is proven.

Now; assuming that we have $n - m$ orthogonal eigenvectors of A

$$v_1, v_2, ..., v_{n-m},\ m < n$$

we want to prove that we have at least one more eigenvector.

Let the vectors $u_1, u_2, ..., u_m$ be orthogonal to both each other and the vectors $v_1, v_2, ..., v_{n-m}$ (which is always possible as we have n vectors in $\mathbb{R}^n$). Additionally, let $U \in \mathbb{R}^{n \times m}$ be the matrix with $u_1, u_2, ..., u_m$ as columns.

Then the matrix $Y = U^T AU$ is a symmetric matrix as

$$Y = U^T AU = (U^T A^T U)^T = \left\{ A = A^T \right\} = (U^T AU)^T$$

Thus, by the same arguments made previously in the base case does Y have at least one eigenvector w with a corresponding eigenvalue $\mu \in \mathbb{R}$ and the following holds

$$Yw = U^T AUw = \mu w \tag{5}$$
$$\Rightarrow UU^T AUw = U\mu w \tag{6}$$

Can now let $v_{n-m+1} = Uw$ which is a linear combination of the vectors in U which implies that v_{n-m+1} is orthogonal to $v_1, v_2, ..., v_{n-m}$ since all vectors $u_1, u_2, ..., u_m$ are so. Substituting this into equation (6) yields

$$\Rightarrow UU^T AUw = U\mu w \tag{7}$$
$$\Rightarrow UU^T Av_{n-m+1} = \mu v_{n-m+1} \tag{8}$$

We can now use that v_{n-m+1} is orthogonal to $v_1, v_2, ..., v_{n-m}$ and show that Av_{n-m+1} is also orthogonal to $v_1, v_2, ..., v_{n-m}$ by

$$(Av_{n-m+1})^T v_i = v_{n-m+1}^T A^T v_i = \left\{ A = A^T \right\} = v_{n-m+1}^T (Av_i) \tag{9}$$

$$= \lambda_i v_{n-m+1}^T v_i = 0 \tag{10}$$

and thus is Av_{n-m+1} also orthogonal to $v_1, v_2, ..., v_{n-m}$ which means that Av_{n-m+1} is also a linear combination of $u_1, u_2, ..., u_m$ and can be written as $Av = Ux$. Substituting this into equation (8) yields

$$UU^T Av_{n-m+1} = UU^T Ux = Ux = Av_{n-m+1} = \mu v_{n-m+1} \tag{11}$$

and thus is v_{n-m+1} an eigenvector of A which which completes the induction and we have proven that a real symmetric matrix has n orthogonal eigenvectors.

Now we just need to prove that it can be decomposed as $A = V\Lambda V^T$. First we will show that it can assumed that all eigenvectors are normalised without loss of generality since

$$Av = \lambda v \Rightarrow A\frac{v}{\|v\|} = \lambda\frac{v}{\|v\|}$$

Now suppose that $v_1, v_2, ..., v_n$ are orthonormal eigenvectors of A with corresponding eigenvalues $\lambda_1, \lambda_2, ..., \lambda_n$ and let V be the matrix with $v_1, v_2, ..., v_n$ as columns. Then the product

$$(V^T V)_{ij} = v_i^T v_j = \begin{cases} 1, & i = j \\ 0, & \text{otherwise} \end{cases}$$

which yields that

$$V^T V = I \Rightarrow V^T V V^{-1} = V^{-1} \Rightarrow V^T = V^{-1}$$

which implies that V is orthogonal.

Letting $\Lambda = \text{diag}(\lambda_1, \lambda_2, ..., \lambda_n)$ we want to show that $A = V\Lambda V^T$. Since $v_1, v_2, ..., v_n$ are orthonormal any vector $x \in \mathbb{R}^n$ can be expressed as

$$x = \sum_{i=1}^{n} \alpha_i v_i$$

Then

$$Ax = A\sum_{i=1}^{n} \alpha_i v_i = \sum_{i=1}^{n} \alpha_i A v_i = \sum_{i=1}^{n} \alpha_i \lambda_i v_i$$

and

$$\begin{aligned} V\Lambda V^T x &= V\Lambda V^T \sum_{i=1}^{n} \alpha_i v_i = V\Lambda \sum_{i=1}^{n} \alpha_i e_i \\ &= V\sum_{i=1}^{n} \alpha_i \lambda_i e_i = \sum_{i=1}^{n} \alpha_i \lambda_i v_i \end{aligned}$$

which yields that

$$Ax = \sum_{i=1}^{n} \alpha_i \lambda_i v_i = V\Lambda V^T x \Rightarrow A = V\Lambda V^T \tag{12}$$

which was to be proven.

Part (iii): *Prove that a positive semi definite matrix has non negative eigenvalues*

Let B be a complex $n \times n$ positive semi-definite matrix accompanied by an eigenvalue λ and an associated eigenvector v. By the definition of a positive semi-definite matrix we know that for any vector $z \in \mathbb{C}$ it holds that $\overline{z}^T Bz \geq 0$. The following thus holds

$$\overline{v}^T Bv = \overline{v}^T \lambda v = \lambda \overline{v}^T v = \lambda \|v\|^2 \geq 0 \tag{13}$$

where $\|v\|^2$ is necessarily positive since eigenvectors are by definition non-zero. This implies that $\lambda \geq 0$, which was to be proven.

Part (iv): *Let $A \in \mathbb{R}^{n\times n}$ symmetric and positive semi-definite matrix. Define a matrix $D = \{D | D_{ij} = A_{ii} + A_{jj} - 2A_{ij}\}$. Show that there exists n vectors $v_1, ..., v_n$, $v_i \in \mathbb{R}^n$ $\forall i$* such that $D_{ij} = \|v_i - v_j\|_2^2$.

$$D_{ij} = \|v_i - v_j\|_2^2 = v_i^T v_i + v_j^T v_j - 2v_i^T v_j \tag{14}$$

Thus, if we can show that any matrix $A \in \mathbb{R}^{n\times n}$ that fulfils the given conditions, can have each of its elements expressed as a dot product between n given vectors $v_1, ..., v_n$, $v_i \in \mathbb{R}^n$ $\forall i$ we have shown what is asked.

Given that A is symmetric and positive semi-definite it can be decomposed into the following eigen-decomposition $A = Q\Lambda Q^T$ where Λ is a diagonal matrix of real eigenvalues (since A is PSD) and Q is an orthogonal matrix of eigenvectors of A. The decomposition can then be rewritten in the following manner

$$A = Q\Lambda Q^T = (\Lambda^{\frac{1}{2}} Q^T)^T (\Lambda^{\frac{1}{2}} Q^T) = V^T V \tag{15}$$

If we thus choose every $v_i,\ i = 1, ..., n$ to be the i:th column in the matrix $(\Lambda^{-\frac{1}{2}} Q^T)$ we get that $A_{ij} = (V^T V)_{ij} = v_i^T v_j$ which yields the final result

$$D_{ij} = A_{ii} + A_{jj} - 2A_{ij} = v_i^T v_i + v_j^T v_j - 2v_i^T v_j = \|v_i - v_j\|_2^2 \tag{16}$$

which was to be proven.

Problem 2

Given that we want to do PCA using k components on a matrix $Y \in \mathbb{R}^{p\times n},\ p \leq n$ where the columns correspond to data points, we can decompose Y in the following manner using SVD.

$$Y = U\Sigma V^T \tag{17}$$

Where U contains the left singular vectors of Y and V contains the right singular vectors of Y.

The transformation matrix W is then given by the first k columns of the matrix U.

If we instead want to perform PCA using k components on Y^T we can use that

$Y^T = (U\Sigma V^T)^T = V\Sigma U^T$

The transformation matrix $\widetilde{W}$ can thus be chosen as the first k columns in V.

Thus is a single SVD computation sufficient for computing PCA on both columns and rows.

Problem 3

Want to maximise the expression $tr(Y^T W W^T Y)$ where $Y \in \mathbb{R}^{d\times n}$, $W \in \mathbb{R}^{d\times k},\ k < d$ under the condition that W has orthonormal columns which implies that $W^TW = I_k$. In order to achieve this we want to use Lagrange multipliers, and for mathematical convenience we will use the cyclic property of trace to shift around the matrices in the expression in the following manner.

$$tr(Y^TWW^TY) = tr(YY^TWW^T) = tr(W^TYY^TW) \tag{18}$$

We thus want to maximise $tr(W^TYY^TW)$ subject to $W^TW - I_k = 0$ which yields the following Lagrangian function

$$\begin{aligned} L &= tr(W^TYY^TW) - \sum_{i=1}^{k}\sum_{j=1}^{k} \lambda_{ij}(w_i^Tw_j - \mathbf{1}\{i=j\}) \quad (19) \\ &= \sum_{i=1}^{k} w_i^TYY^Tw_i - \sum_{i=1}^{k}\sum_{j=1}^{k} \lambda_{ij}(w_i^Tw_j - \mathbf{1}\{i=j\}) \quad (20) \end{aligned}$$

where $\mathbf{1}$ is the indicator function.

Taking the derivative with respect to the λ_{lj} only results in the initially stated condition that

$$W^TW = I_k \tag{21}$$

Taking the derivative with respect to $w_l,\ l = 1, ..., k$ yields

$$\frac{\partial L}{\partial w_l} = 2YY^Tw_l - \sum_{j=1}^{k}(\lambda_{lj}w_j + \lambda_{jl}w_j) = 0 \tag{22}$$

By letting Λ be a matrix with elements $(\Lambda)_{lj} = \lambda_{lj}$ equation (22) can be expressed using matrices for all $l = 1, ..., n$.

$$2YY^TW - W(\Lambda^T + \Lambda) = 0 \Rightarrow YY^TW = W\,\frac{\Lambda^T + \Lambda}{2} \tag{23}$$

Noting that $\frac{\Lambda^T+\Lambda}{2}$ is trivially symmetric it can be diagonalised in the following manner: $\frac{\Lambda^T+\Lambda}{2} = PDP^{-1}$. Substituting this into equation (23) results in

$$YY^TW = W\frac{\Lambda^T+\Lambda}{2} = WPDP^{-1} \tag{24}$$
$$\Rightarrow W^TYY^TW = W^TWPDP^{-1} = \left\{W^TW = I_k \text{ by } (21)\right\} = PDP^{-1} \tag{25}$$
$$\Rightarrow P^{-1}W^TYY^TWP = P^{-1}PDP^{-1}P \tag{26}$$
$$\Rightarrow P^{-1}W^TYY^TWP = D \tag{27}$$

Thus is YY^T diagonalised by WP and D thus consist of k eigenvalues of YY^T and by substituting equation (26) into the original trace expression we get

$$tr(W^TYY^TW) = tr(PDP^{-1}) = tr(P^{-1}PD) = tr(D) = \sum_{i=1}^{k} d_i \tag{28}$$

where d_i are eigenvalues of YY^T. Thus in order to maximise this we just choose $d_i, i = 1, 2, ..., k$ to be the k largest eigenvalues of YY^T.

Now to show that if we select $W = U_k$ we get the same maximum value. Using the skinny SVD of $Y = U\Sigma V^T$ and substituting into the original trace expression we get

$$\begin{aligned}
tr(Y^TWW^TY) &= tr(V\Sigma^TU^TWW^TU\Sigma V^T) && (29)\\
&= tr(V^TV\Sigma U^TWW^TU\Sigma) = tr(\Sigma U^TWW^TU\Sigma) && (30)\\
&= tr(\Sigma U^TU_kU_k^TU\Sigma) = tr((\Sigma U^TU_k)(\Sigma U^TU_k)^T) && (31)\\
&= tr((\Sigma I_{d\times k})(\Sigma I_{d\times k})^T) = tr\Sigma^2_{kxk} && (32)\\
&= \sum_{i=1}^{k}\sigma_i^2 = \left\{k \text{ largest singular valies}\right\} = \sum_{i=1}^{k} d_i && (33)
\end{aligned}$$

Thus is $tr(W^TYY^TW)$ subject to $W^TW - I_k = 0$ maximised by choosing $W = U_k$ and the maximum value is given by $\sum_{i=1}^{k}\sigma_i^2$.

Problem 4

In order to show that it provides a correct estimation of the Gram matrix S we need to prove that we can derive a similarity matrix S from a matrix $D$where$(D)_{ij} = d_{ij} = (z_i - z_j)^T(z_i - z_j)$ such that

1. $S = Z^T Z$

2. Z gives 0 reconstruction error for the matrix D

Normally $s_{ij} = y_i T y_j$, we claim that it is valid to express

$$s_{ij} = -\frac{1}{2}(d_{ij}^2 - d_{1j}^2 - d_{1i}^2) \tag{34}$$

where

$$d_{ij}^2 = (y_i - y_j)^2 = y_i^2 + y_j^2 - 2y_i y_j \tag{35}$$

.

Substituting (35) into (34) yields the following

$$\begin{aligned} s_{ij} &= -\frac{1}{2}(d_{ij}^2 - d_{1j}^2 - d_{1i}^2) && (36) \\ &= -\frac{1}{2}(y_i^2 + y_j^2 - 2y_i^T y_j - y_1^2 - y_i^2 + 2y_1^T y_i - y_1^2 - y_j^2 + 2y_1^T y_j) && (37) \\ &= -\frac{1}{2}(-2y_i^T y_j - 2y_1^2 + 2y_1^T y_i + 2y_1^T y_j) && (38) \\ &= (y_i - y_1)^T (y_j - y_1) = z_i^T z_j && (39) \end{aligned}$$

Let T be a matrix of the same size as Y, with all columns as y_1. Then the Gram matrix S can be written as

$$S = (Y - T)^T (Y - T) = Z^T Z \tag{40}$$

Thus is 1. shown. Now we want to show 2.

We know that $d_{i,j} = (z_i - z_j)^T (z_i - z_j)$ and by substituting the result from equation (39) we get the following

$$\begin{aligned} d_{ij}^2 &= (z_i - z_j)^T (z_i - z_j) && (41) \\ &= (y_i - y_1 - (y_j - y_1))^T (y_i - y_1 - (y_j - y_1)) && (42) \\ &= (y_i - y_j)^T (y_i - y_j) && (43) \end{aligned}$$

Which shows that Z gives 0 reconstruction error for the distance matrix D.

Problem 5

We are considering the classical MDS when $Y \in \mathbb{R}^{d \times n}$ is known which implies that we can construct a similarity matrix $S = Y^T Y$. An MDS embedding can then be obtained by performing eigen-decomposition on S which yields the following

$$S = Y^T Y = Q \Lambda Q^T = (\Lambda^{-\frac{1}{2}} Q^T)^T (\Lambda^{-\frac{1}{2}} Q^T) \tag{44}$$

where Λ is a diagonal matrix with eigenvalues of $S = Y^T Y$ sorted in descending order.

The latent matrix X can then be chosen as

$$X = I_{k \times n} \Lambda^{-\frac{1}{2}} Q^T, \; k < d \tag{45}$$

However, this can also be written in terms of the SVD of $Y = U \Sigma V^T$

$$\begin{aligned} S = Y^T Y = V \Sigma^T U^T U \Sigma V^T = V \Sigma^T \Sigma V^T = (\Sigma V^T)^T (\Sigma V^T) \quad &(46) \\ = (\Lambda^{-\frac{1}{2}} Q^T)^T (\Lambda^{-\frac{1}{2}} Q^T) \quad &(47) \end{aligned}$$

Since Λ and Σ has ordered diagonals by magnitude and V contains the right singular vectors we know that $V = Q$ which gives that

$$\Lambda^{-\frac{1}{2}} Q^T = \Sigma V^T \tag{48}$$

Thus can the latent matrix also be expressed as

$$X = I_{k \times n} \Sigma V^T, \; k < d \tag{49}$$

In PCA we decompose Y into its SVD, yielding

$$Y = U \Sigma V^T \tag{50}$$

The latent matrix is then chosen as

$$\begin{aligned} X &= (U I_{n \times k})^T Y = I_{k \times n} U^T Y \quad &(51) \\ &= \left\{ Y = U \Sigma V^T \Rightarrow U^T Y = \Sigma V^T \right\} \quad &(52) \\ &= I_{k \times n} \Sigma V^T, \; k < d \quad &(53) \end{aligned}$$

Comparing equations (49) and (53) we see that the results are equivalent.

Regarding the computational efficiency I believe that there are two areas to consider: computational complexity and numerical accuracy. The complexities of computing the SVD and eigen-decomposition are similar and somewhat dependant on the properties of the input matrix, eigen-decomposition often performing a bit better. However, for the MDS we are required to perform a matrix multiplication beforehand which both introduces additional cost in time but can also introduce issues with numerical accuracy causing the resulting eigen-decomposition to have imaginary parts. For this reason the SVD should be preferred.

Problem 6

We want to argue that the process to obtain a neighbourhood graph G in the Isomap method may yield a disconnected graph. Given that we have a set of points that corresponds to those visualised in 1 the process to obtain a neighbourhood graph G will yield a disconnected graph if we only connect a point to its 2 nearest neighbours. This as for any given point in cluster A, its two nearest neighbours both belong to cluster A and for any given point in cluster B, its two nearest neighbours both belong to cluster B. Thus are cluster A and B not connected and the graph is disconnected, causing the Isomap algorithm to fail.

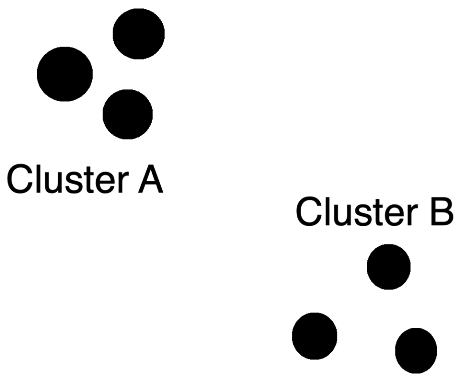

Figure 1: Example of 2 clusters

An example of heuristic to path this problem is to use create artificial data points that bridge the gap between the different clusters, resulting in the graph to become connected again. This could be achieved by using the fact that a disconnected neighbourhood graph implies that there exists clusters within the data and use the following scheme:

1. Identify all $C_i,\ i = 1, 2, ..., k$ clusters by using the neighbourhood function G
2. Pick a random point $p_i,\ i = 1, 2, ..., k$ of each cluster (the mean of the cluster would also work but harder to justify)
3. Connect all points p_i with each other using linspace(p_i, p_j, n) where n is chosen such that the distance between the artificial data points is equal to the smallest distance of the original data set.

This would ensure that all clusters would be tied together as the distances between the artificial points equals the smallest distance of the original data set, ensuring that the artificial points will always belong to the neighbourhood sets of each other. The expected value of the distance between two clusters using this method will roughly correspond to the distance between the two cluster means.

Problem 7

The goal of this problem is to visualise how similar different animals at a zoo are by projecting the given data from $\mathbb{R}^{16}$ to $\mathbb{R}^2$ using three different embeddings; PCA, MDS and Isomap. The data matrix Y will be structured in the same way as in the lecture such that a column of Y corresponds to a data point, thus is $Y \in R^{16\times101}$ as there are 101 data points.

Preprocessing the data

In order to enable the applications of the embeddings the data has to be preprocessed by first removing the columns 'type' and 'animal name' from the data set. These can later be used in the visualisation.

The remaining attributes are all boolean with values in $\{0, 1\}$ except the attribute 'legs' which takes values in $\{0, 2, 4, 6, 8\}$. This attribute thus takes on values up to 8 times the magnitude in comparison to the remaining attributes which results in the legs attribute to be seen as more important for the embeddings. For this reason the magnitude of 'legs' will be scaled down by 8 times, causing it to instead take values in $\{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$ giving it a similar magnitude to the remaining attributes.

Visualisation of data

In order to project the data in 2D such that similar animals are projected close to one another the first two resulting latent variables from a method will be plotted against each other as they contain the most information about the data set as a result of the sorting of the singular values and eigenvalues in descending order.

PCA

The implementation of PCA was solved in the following manner

Algorithm 1: PCA method

```
  Input: Data matrix $Y$
  Output: 2-dimensional embedding $X$
  Data: Zoo animals
1 Center the columns (data points) of the data matrix:
    $Y_c \leftarrow \text{center}(Y)$
2 Compute SVD of $Y_c$: $[U, \Sigma, V] \leftarrow \text{SVD}(Y_c)$
3 Select the first two columns of $U$: $W \leftarrow [u_1, u_2]$
4 Compute embedding: $X \leftarrow W^T Y_c$
```

The implementation was quite short using Numpy's Singular Value Decomposition function. The only concern that arised was the centring of the data matrix as it removes the boolean nature of the matrix. However I argue that this only translates the data points and does not remove the information of the attributes which renders the action viable.

The following visualisation was achieved

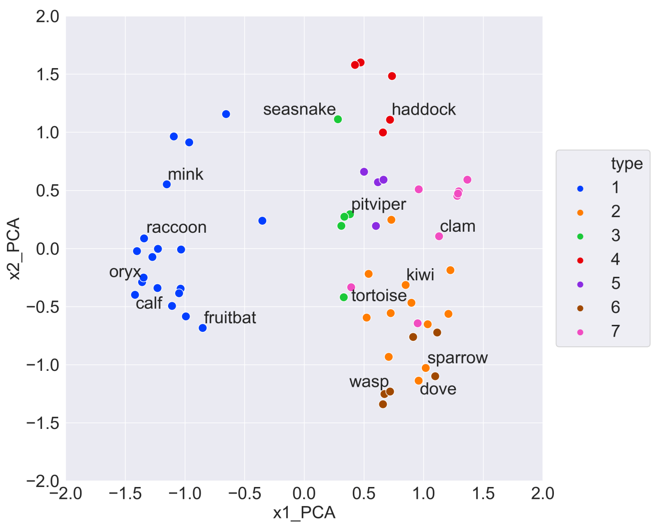

Figure 2: Visualisation PCA

where a few animals are annotated to give some intuition and understanding behind the placing of the different animals. It is apparent from the plot that there is a clear distinction between type 1 and the remaining types $2-7$ whom are more mixed among each other.

MDS

The implementation of MDS was solved in the following manner

Algorithm 2: MDS method

```
    Input: Data matrix $Y$
    Output: 2-dimensional embedding $X$
    Data: Zoo animals
1   Compute a distance matrix of the data: $D \leftarrow \text{weighted\_distance}(Y)$
2   Compute a similarity matrix from $D$: $S \leftarrow \text{similarity\_matrix}(D)$
3   Compute Eigen-decomposition of $S$: $[D, Q] \leftarrow \text{Eig}(S)$
4   Order $D$ in descending order of eigenvalues magnitude and $Q$
      correspondingly
5   Ensure elements of $D$ and $Q$ are real
6   Compute embedding: $X \leftarrow I_{2\times 101} D Q^T$
```

In the implementation of MDS it was proposed to infer the importance of different attributes by taking it into account when computing the pairwise distance. I solved this by replacing the pairwise distance $d(y_i, y_j)$ with $d_w(y_i, y_j)$ where

$$d(y_i, y_j) = \|y_i - y_j\|^2 = (y_i - y_j)^T (y_i - y_j)$$
$$d_w(y_i, y_j) = (y_i - y_j)^T W (y_i - y_j)$$

y_i, y_j are two data points and $W = diag(w_1, w_2, ..., w_k)$ where each w_i is a weight for the corresponding attribute. If $w_i = 1, \forall i$ it is the same distance as before but if one weight, lets say $w_1 = 2$ attribute is twice as important as before since if that attribute differs between y_i, y_j, that distance is multiplied by 2 now.

One concern that arose during the implementation of MDS was that the eigenvalues and eigenvectors had small complex components even though the matrix was symmetric. I believe that this was caused by the numerical imprecision of the computer and solved it by simply removing the complex components.

The following visualisation was achieved

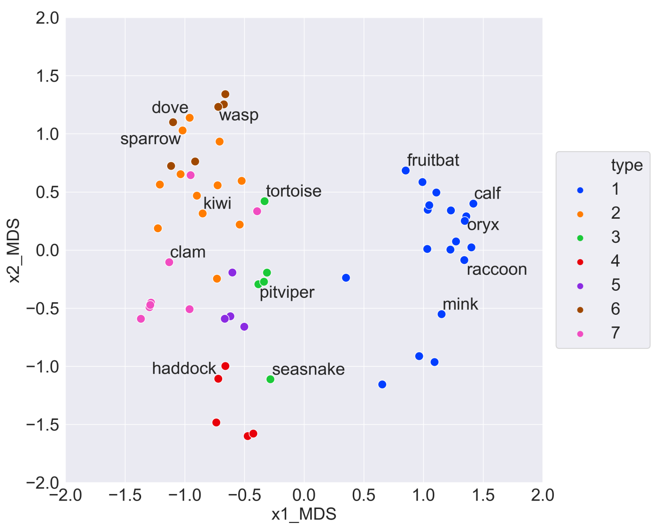

Figure 3: Visualisation MDS, all weights equal to 1

Here one can see that using all weights equal to 1 yields a very similar result to the one of PCA only that the points are mirrored over both the y and x axis which I believe is a result of the eigenvalue algorithm of Numpy since one can change the sign of an eigenvector and it still fulfils the eigenvalue equation.

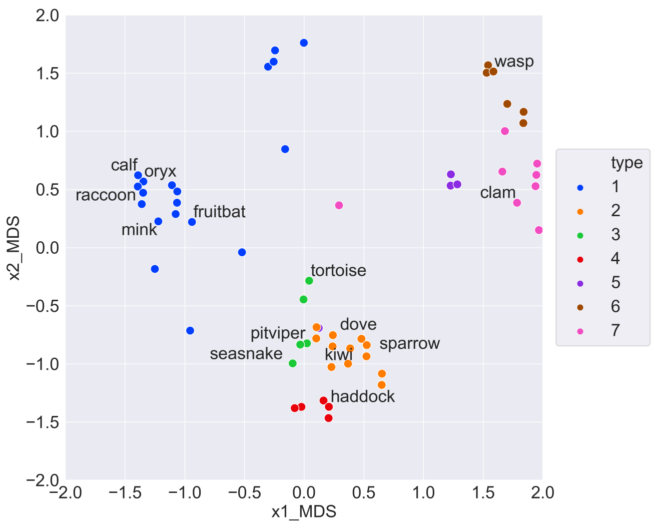

Figure 4: Visualisation MDS, $w_{legs} = 4, w_{tail} = 4$

Here the results become a bit more interesting where I have chosen to put more weight into the number of legs an animal has and whether or not it is an aquatic animal. This has caused a clear sep

Isomap

The implementation of Isomap was solved in the following manner

Algorithm 3: Isomap method

```
    Input: Data matrix $Y$
    Output: 2-dimensional embedding $X$
    Data: Zoo animals
1   Select number of neighbours $k$
2   Compute a graph matrix: $G \leftarrow$ graph_matrix(data matrix, $k$)
3   Compute shortest path matrix: $P \leftarrow$ shortest_path(G)
4   Compute similarity matrix: $S \leftarrow$ similarity_matrix($P$)
5   Compute embedding: $X \leftarrow$ MDS($S$)
```

Argue why you should not use quadratic distance

Comparison

Even though no distance in PCA, could still infer importance by scaling attributes.