Dimension Reduction

Linear Algebra Project Report, Dr.Ramin Javadi

MohammadSadegh Akhoundzadeh*[1] and Maryam Meghdadi†[1]

[1]Department of Electrical and Computer Engineering, Isfahan University of Technology

July 4, 2018

Abstract

In this report we give a review on some methods of dimension reduction. Given some input high-dimensional data, the goal of dimension reduction is to map them to a low-dimensional space such that certain properties of the initial data are preserved. Finally, we will compare discussed methods in some experiments.

1 Introduction

In machine learning classification problems, there are often too many factors on the basis of which the final classification is done. These factors are basically variables called features. The higher the number of features, the harder it gets to visualize the training set and then work on it. Sometimes, most of these features are correlated, and hence redundant. This is where dimensionality reduction algorithms come into play. Dimensionality reduction is the process of reducing the number of random variables under consideration, by obtaining a set of principal variables. It can be divided into feature selection and feature extraction. Some of the dimension reduction methods are based on linear transformation like PCA (Principal Component Analysis) and some of them are based on nonlinear techniques such as Isomap and Spectral Embedding.

2 Theory

2.1 Linear Methods

2.1.1 PCA

Principal component analysis(PCA) is a very popular technique for dimensionality reduction. Given a set of n-dimensional data, PCA algorithm aims to find d-dimensional data ($d < n$) on a linear subspace. Such a reduced subspace attempts to maintain most of the variability of the data. The linear subspace can

*ms.akhondzadeh@gmail.com
†marmegh@gmail.com

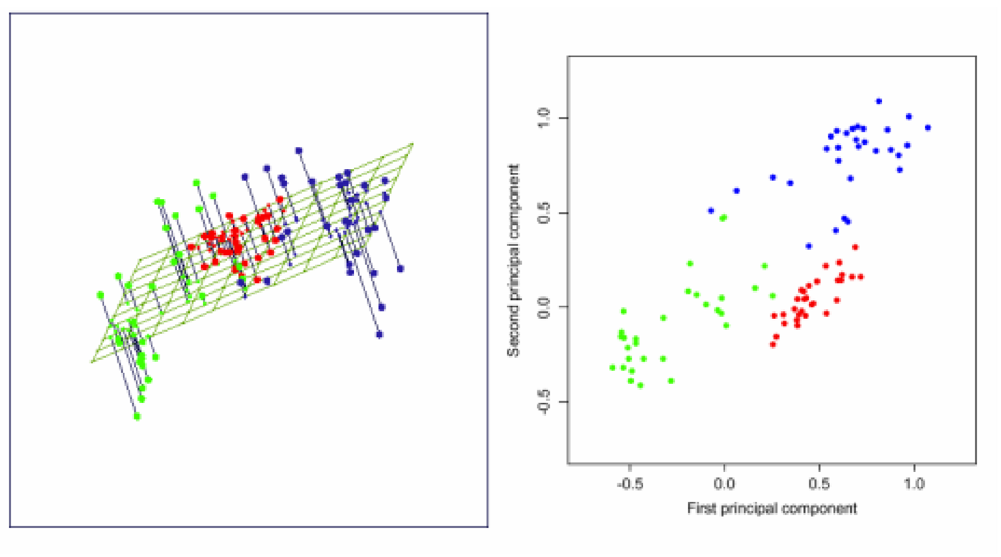

Figure 1: PCA applied to a 3D data set to reduce it to two dimensions

be specified by d orthogonal vectors that form a new coordinate system, called the 'principal components'. The goal is to find some vectors on a subspace which the variation of the data is maximal on it.
Example: In this example we projected a 3D data in a 2D subspace. Actually, we project the data on the plane of two eigenvectors U1 and U2.(Figure 1)

In order to capture as much of the variability as possible, let us choose the first principal component, denoted by U_1, to have maximum variance. Our data is n $\times$ t matrix X and our first eigenvector is a combination of X with ω coefficients such that $\omega = [\omega_1 \dots \omega_n]$. So we have:

$$U_1 = \omega^T X \tag{1}$$

$$\mathrm{var}(U_1) = \mathrm{var}(\omega^T X) = \omega^T S \omega \tag{2}$$

where S is the n $\times$ n sample covariance matrix of X. To reach our goal we should maximize the variance we obtained in (2).

$$\max\ \omega^T \mathrm{S}\ \omega \quad \text{s.t.}\ \omega^T \omega = 1$$

To do this, we use lagrange method and introduce a lagrange multiplier λ:

$$L(\omega, \lambda) = \omega^T S \omega - \lambda(\omega^T \omega - 1) \tag{3}$$

By differentiating L with respect to ω we have:

$$S\omega = \lambda_1 \omega \tag{4}$$

This shows that the first principal component is given by the normalized eigenvector with the largest associated eigenvalue of the sample covariance matrix S.

A similar argument can show that the d dominant eigenvectors of covariance matrix S determine the first d principal components. To obtain these eigenvectors we find singular value decomposition (SVD):

$$X = U \Sigma V^T \tag{5}$$

where columns of U are eigenvectors of XX^T (covariance matrix). We define Y as our projected d-dimensional data (d < n):

$$Y = U(:, 1:d)^T X \qquad or \qquad Y = U_d^T X \tag{6}$$

Next step is to reconstruct the training data which is:

$$\hat{X} = U_d Y \tag{7}$$

For a new test example x we compute $y = U^T x$ and then reconstruct it $\hat{x} = Uy$. [2]

2.1.2 Dual PCA

Dual PCA is similar to PCA but it's using a trick that make this method faster in the case that the number of features is bigger than the samples. The goal is to reduce dependence of our algorithm on n (the number of features). So instead of decomposition of XX^T, we decompose $X^T X$. Note that in the SVD factorization $U \Sigma V^T$, so $XV = U\Sigma$, where the eigenvectors in U corresponds to nonzero singular values in Σ. Thus, the following conversion between the top d eigenvectors can be derived:

$$U = XV\Sigma^{-1} \tag{8}$$

Replacing all uses of U in PCA algorithm gives us Dual PCA algorithm. First we compute Y:

$$Y = U^T X = \Sigma V^T \tag{9}$$

Then, reconstruct data:

$$\hat{X} = UY = U\Sigma V^T = XV\Sigma^{-1}\Sigma V^T = XVV^T \tag{10}$$

Next, project out of sample point x:

$$y = U^T x = \Sigma^{-1} V^T X^T x \tag{11}$$

And finally reconstruct this sample point:

$$\hat{x} = Uy = UU^T x = XV\Sigma^{-2} V^T X^T x \tag{12}$$

2.1.3 Random Projection

Again, if the number of samples is large, we can use a random method which guarantees that after projecting the points from the n-dimensional space to K dimensional space using the randomly drawn matrix W, the distances between the high dimensional points is preserved in the lower dimensional projections up to some approximation factor. This matrix W is:

$$W[i,j] = \left\{ \begin{array}{ll} +\frac{1}{\sqrt{K}} & \text{with probability } 1/2 \\ -\frac{1}{\sqrt{K}} & \text{with probability } 1/2 \end{array} \right\}$$

2.2 Nonlinear Methods

2.2.1 Isomap

The euclidean distance of points on a linear space remains unchanged but on a submanifold (nonlinear space) this distance will change. This method is similar to MDS but we change euclidean distance to geodesic distance. The geodesic distances represent the shortest paths along the curved surface of the manifold measured as if the surface were flat. This can be approximated by a sequence of short steps between neighboring sample points. Isomap then applies MDS to the geodesic rather than straight line distances to find a low-dimensional mapping that preserves these pairwise distances. Steps of Isomap are:

1. Construct a k-nearest neighbor graph on n data points (connect each point to it's k-nearest neighbor.
2. Compute shortest path between all the points (D^G) as an estimation of geodesic distance, using Dijkstra's algorithm and Floyd-Warshall's algorithm.
3. Compute $k = -\frac{1}{2}H(D^G)^2H$. find eigenvectors of k and call it V. then find the top p eigenvalues of k and form $\hat{\Lambda}$. The final solution is $Y = \hat{\Lambda}^{\frac{1}{2}}V^T$. Note that matrix H is a centering matrix and is equal to $H = I - \frac{1}{n}ee^T$, where e is a n $\times$ 1 vector of ones and $\hat{\Lambda}$ is a diagonal matrix which diagonal values are the top p eigenvalues of X^TX.

2.2.2 Laplacian Eigenmaps (Spectral Embedding)

Given t points in n-dimensional space, the Laplacian eigenmaps Method (LEM) starts by constructing a weighted graph with t nodes and a set of edges connecting neighbouring points. The neighbourhood graph can be constructed by finding the k nearest neighbours. For weighting the edges, we can use adjacency matrix, which elements are one for the nodes that are connected and zero for disconnected nodes. Another way is to use gaussian distance:

$$W_{ij} = e^{-\frac{||x_i - x_j||^2}{\gamma}} \tag{13}$$

The embedding map is then provided by the following objective:

$$\min_Y \sum_{i=1}^{t}\sum_{j=1}^{t}(y_i - y_j)^2 W_{ij} \tag{14}$$

subject to appropriate constraints. From spectral clustering we know that:

$$\sum_{i,j}(y_i - y_j)^2 W_{ij} = Y^T LY \tag{15}$$

So From (14) and (15) we reach to:

$$\min_Y Tr(YLY^T) \tag{16}$$

$$\text{s.t. } YY^T = I$$

Where L is laplacian matrix which is $L = D - W$ and D is diagonal and equals to $D_{ii} = \sum_{j=1}^{t} W_{ij}$. From (13) and (15) we conclude that if x_i and x_j are far apart then W_ij becomes small, so we don't care about y_i and y_j. But if x_i and x_j are close, W_ij becomes large, so y_i and y_j should be small and we reach to our goal.

3 Experiment

We start by generating three different data to compare the algorithms.

3.1 Part1

At first we create data in the shape of "hello"(figure 2).

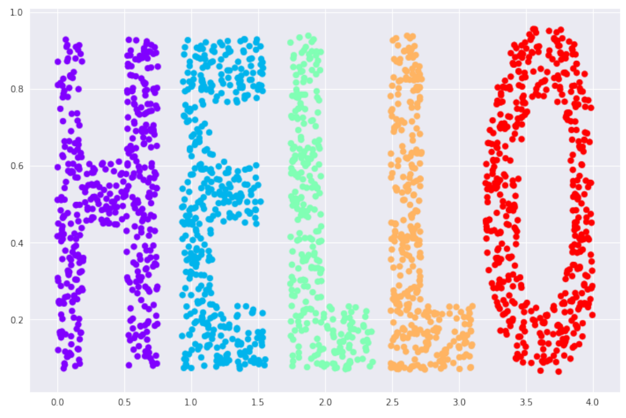

Figure 2: hello-shaped data

Then for the first part of our experiments we use a linear function with a gaussian noise to add a new dimension to the data and make it 3D. (figure 3)

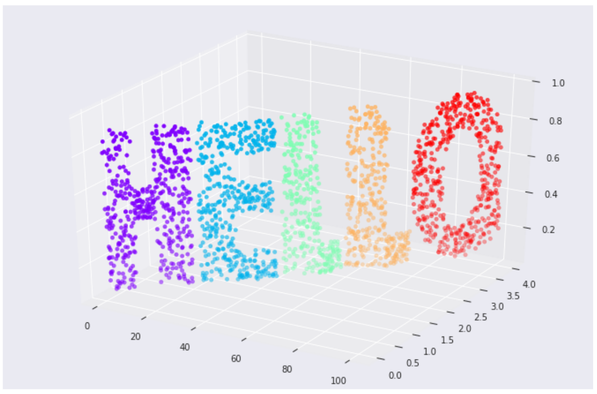

Figure 3: 3D hello-shaped data

See the result of the the algorithms below:

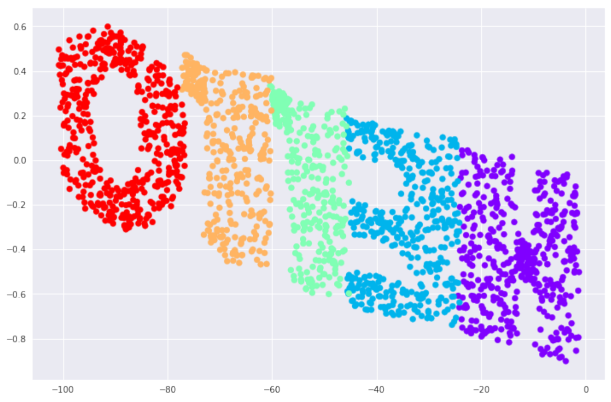

Figure 4: PCA applied to 3D hello(figure 3) set to reduce it to two dimensions

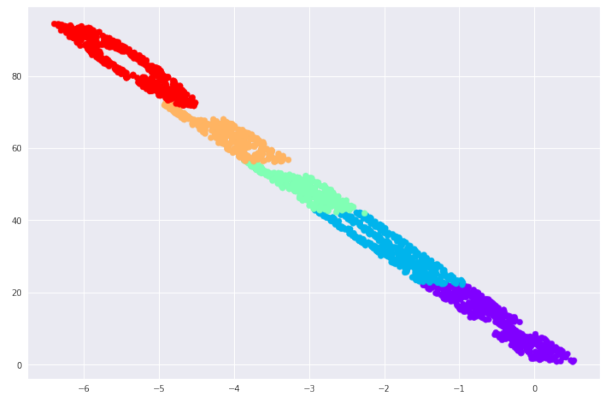

Figure 5: Random Projection applied to 3D hello(figure 3) set to reduce it to two dimensions

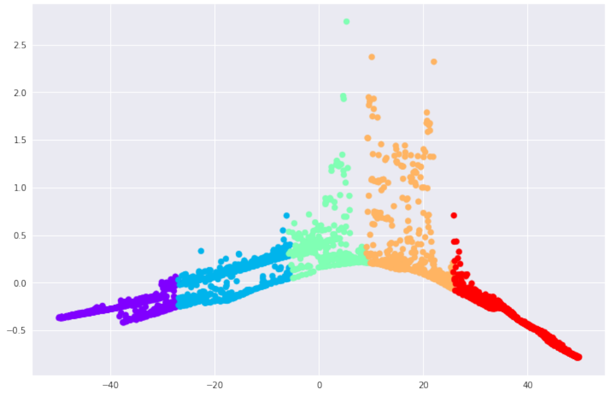

Figure 6: Isomap applied to 3D hello(figure 3) set to reduce it to two dimensions

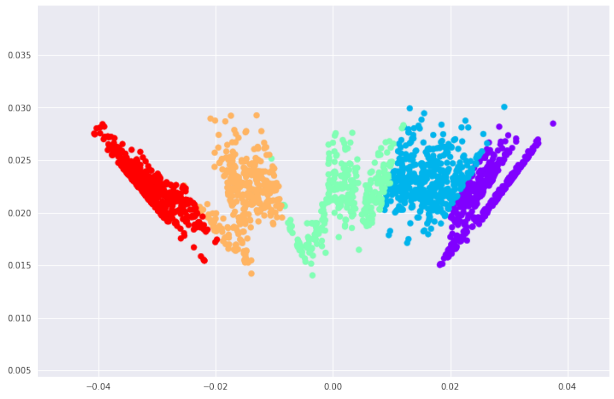
Figure 7: Laplacian applied to 3D hello (figure 3) set to reduce it to two dimensions

3.2 Part2

In the next part instead of adding a new dimension to the data, we maked the "hello" data curved (figure 8)

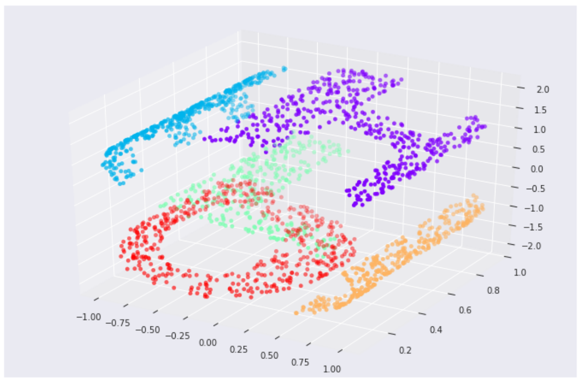
Figure 8: curved hello

so now we have a manifold in the shape of "hello" in three dimensions and our goal is to reduce the dimension while preserving the certain properties of the data.

See the result of the the algorithms below:

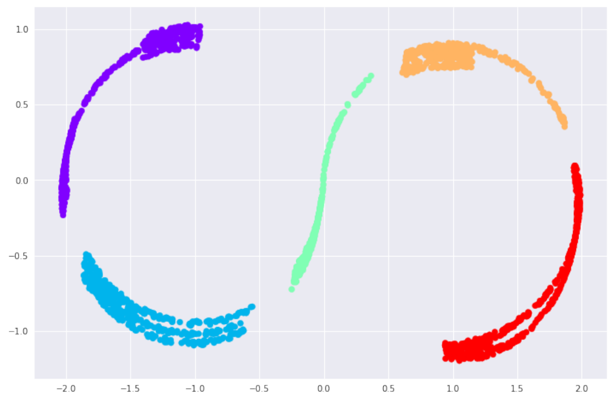

Figure 9: PCA applied to curved hello (figure 8) set to reduce it to two dimensions

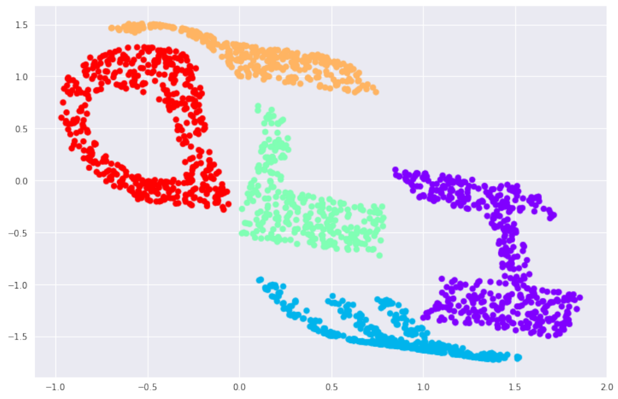

Figure 10: Random Projection applied to curved hello (figure 8) set to reduce it to two dimensions

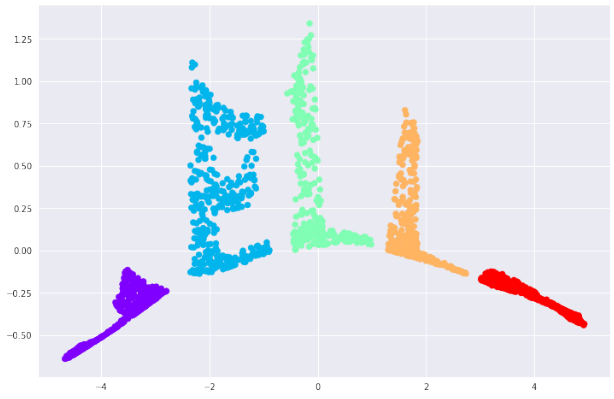

Figure 11: Isomap applied to curved hello (figure 8) set to reduce it to two dimensions

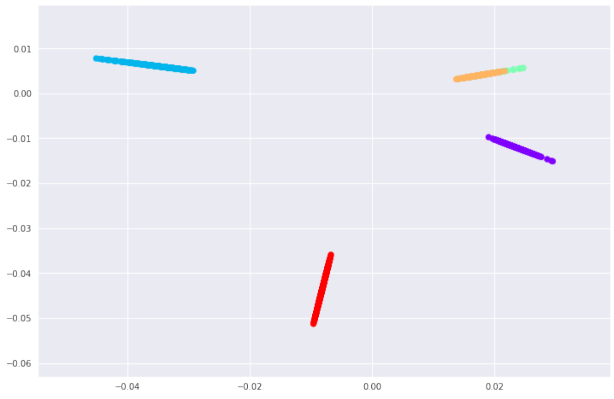

Figure 12: Laplacian applied to curved hello(figure 8) set to reduce it to two dimensions

3.3 Part3

In the last part, we use a S-shaped data which is curved again. we repeat previous section with this new data.

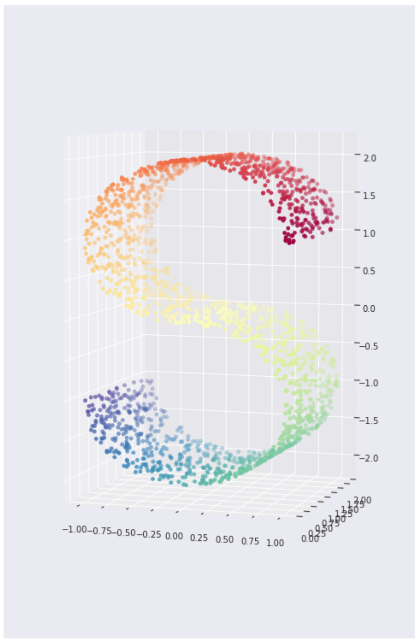

Figure 13: S-shaped curve

See the result of the the algorithms below:

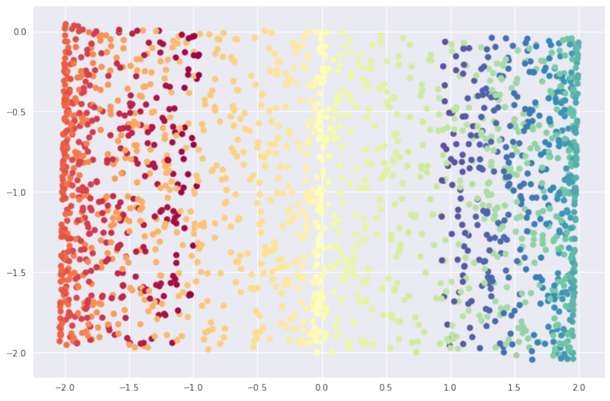

Figure 14: PCA applied to S-shaped (figure 13) set to reduce it to two dimensions

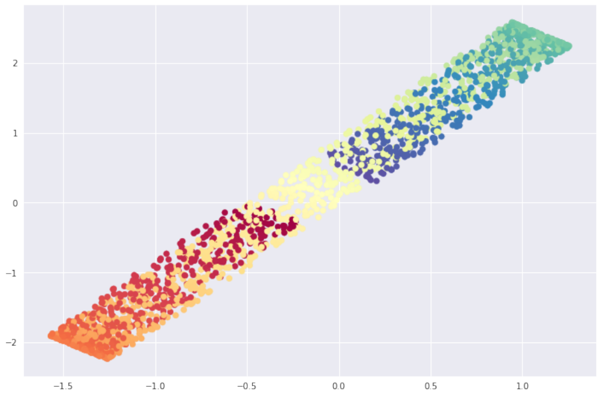

Figure 15: Random Projection applied to S-shaped (figure 13) set to reduce it to two dimensions

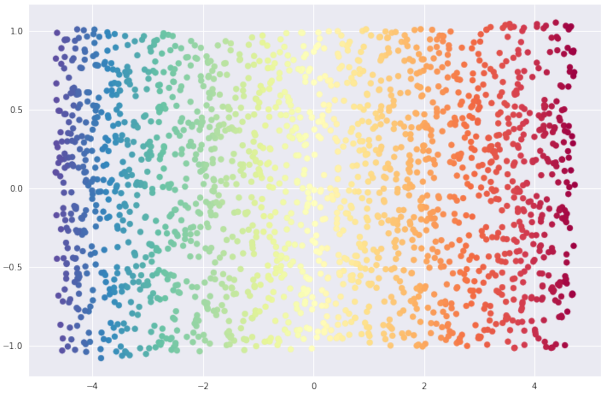

Figure 16: Isomap applied to S-shaped (figure 13) set to reduce it to two dimensions

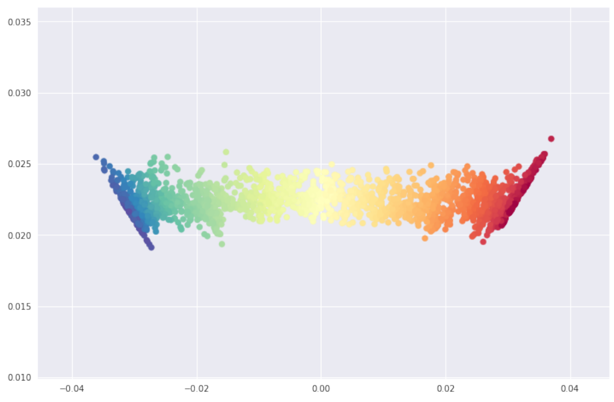

Figure 17: Laplacian applied to S-shaped (figure 13) set to reduce it to two dimensions

[2] [1] [3]

References

[1] Ali Ghodsi. Dimensionality Reduction A Short Tutorial. *Science*, page 25, 2006.

[2] E Kokiopoulou, J Chen, and Y Saad. Trace Optimization and eigenproblem in dimension reduction methods. *Numerical Linear Algebra with Applications*, pages 1–35, 2011.

[3] Random Projections. Machine Learning for Data Science (CS 4786) Why Random Projection Works ?! (Cs 4786):1–5.