FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Analysis of hubness and application of reduction methods on high dimensional datasets

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Wirtschaftsinformatik

eingereicht von

Roscoe Baston
Matrikelnummer 0825533

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Ao.univ.Prof. Dr. Andreas Rauber
Mitwirkung: Dipl. Ing. Thomas Lidy

Wien, 3.12.2015

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Analysis of hubness and application of reduction methods on high dimensional datasets

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Business Informatics

by

Roscoe Baston
Registration Number 0825533

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao.univ.Prof. Dr. Andreas Rauber
Assistance: Dipl. Ing. Thomas Lidy

Vienna, 3.12.2015

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Roscoe Baston
Reinprechtsdorfer Strasse 39/17, 1050 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Acknowledgements

I would like to express my sincere gratitude to my advisor Andreas Rauber for his support, motivation and knowledge he provided for my master thesis. His guidance helped me during the research and writing of this thesis.

Secondly, I would like to thank Thomas Lidy and Abdel Aziz Taha for the support they provided.

I could not have had better advisors and mentors.

Last but not the least, I would like to thank my family, friends and colleagues for supporting me throughout writing this thesis and life in general.

Abstract

Hubness is a fairly new issue emerging out of the domains business intelligence and machine learning, that originates from an asymmetric distance relationship between two points inside a dataset. In a high dimensional dataset with a high number of points, this will result is a small amount of points with a lot of neighbors with fairly short distance, the so called “hub points”, and many points with long distance to all neighbors, the “anti hubs”. That’s the reason why hub points occur frequently as neighbors to other points. On the other hand, most other points rarely occur as neighbors and therefore have little influence on subsequent classification.

There is a Matlab hubness toolbox which is able to calculate the hubness of a dataset, however, it does not contain functions to deal with large datasets. Also, neither a method to calculate the distance matrix using different metrics and nor a method to compare those hubness values are implemented in the toolbox. Those functions were added by myself in the course of the programming work for this thesis.

This thesis aims to answer research questions regarding the contained hubness of datasets and the possibility to reduce said hubness using new reduction methods while verifying that the quality of the data is not reduced. Two of the reduction methods proposed in this thesis are the exclusion of the biggest hubpoints and the projection on a hypersphere both proposed by Abdel Aziz Taha. The third reduction method is the execution of a principal component analysis to find the most important dimensions and visualize the hub points and the data in form of a plot. After the implementation of these new reduction methods, and the application on artificial and non-artificial data, it became clear that the reduction methods operate as expected, reducing the hubness. The pca reduction method turned out to be a non-viable approach though, since the quality of the data and the retrieval system was reduced in the process. The other two reduce hubness and do not damage the quality of the data and the retrieval system.

Another general observation was the fact that artificial datasets tended to contain more hubness than non-artificial datasets.

Kurzfassung

Hubness ist ein relativ neues Problem, welches aus den Sparten Business Intelligence und Machine Learning kommt. Es entsteht durch ein asymmetrisches Distanzverhältnis zwischen zwei Punkten in einem Datenset. Dieses Verhältnis ist der Grund, warum es wenige Punkte mit vielen Nachbarn gibt, welche alle kurze Distanzen zueinander haben. Diese werden “hub points” genannt. Auf der anderen Seite gibt es viele Punkte mit großer Distanz zu deren Nachbarn. Diese werden als “anti hubs” bezeichnet. Das ist der Grund warum Hub Punkte oft als Nachbarn anderer Punkte gefunden werden und warum viele andere selten als Nachbarn gefunden werden. Diese haben dann auch wenig Einfluss auf spätere Klassifikation.

Es gibt bereits eine Matlab hubness Toolbox, welche Methoden zur Hubnessmessung anbietet. Es fehlt jedoch die Möglichkeit die Hubness für große Datenmengen zu berechnen. Außerdem beinhaltet sie keine Möglichkeit unterschiedliche Metriken zu verwenden und diese zu vergleichen. Diese Funktionen wurden von mir im Zuge der Programmierarbeit für die Diplomarbeit implementiert.

Diese Diplomarbeit versucht Forschungsfragen bezüglich beinhalteter Hubness mit diversen Metriken (L1, L2, Linf, cosine) zu beantworten. Außerdem soll die Hubness mittels neuer Methoden reduziert werden. Dabei sollte die Datenqualität nicht reduziert werden. Zwei dieser Reduktionsmethoden, die Exclusion der größten Hub Punkte und die Projektion auf eine Hypershpere wurden von Abdel Aziz Taha vorgeschlagen. Die dritte Reduktionsmethode ist eine principal component analysis welche dazu eingesetzt wird die wichtigsten Dimensionen zu finden. Gleichzeitig kann diese verwendet werden um die Daten und die Hubpunkte zu visualisieren. Diese Methoden wurden unter Verwendung von generierten Daten implementiert und getestet. Dies hat gezeigt, dass alle Methoden erwartungsgemäß funktionieren und die Hubness reduzieren. Allerdings stellte sich heraus das die pca Methode keine brauchbare Reduktionsmethode darstellt, weil dabei die Datenqualität reduziert wird. Eine weitere Beobachtung war, dass generierte Daten tendenziell mehr Hubness beinhalten als “real-world” Daten.

Contents

CHAPTER 1

Introduction

1.1 General Information

Science evolved from Experimental Science, which only observed phenomena, to Data-Intensive Science, which connects theory, experiment and simulation. The data is generated from many different sources (instruments, simulations, sensor networks, ...) and increasingly gains value. It needs to be stored, analyzed and turned into knowledge. [1] This transformation from data to information is called "Business Intelligence". The implicit assumption that insights are gained out of data are correct. Also, data is the core of almost every domain today, and its value is increasingly recognized. Therefore, many business decisions are based on machine learning.

Machine learning is defined as a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can teach themselves to grow and change when exposed to new data.

Researchers have used machine learning algorithms to solve hard problems in a variety of domains, enabling exciting, new applications of computing. A typical classification pipeline must first collect raw data, which is then processed to feature vectors. These are then used to train a model and finally evaluated in experiments. [2]

Machine learning has a well known phrase "curse of dimensionality", which refers to difficulties that arise when analyzing datasets that have high dimensionality. New issues become more prominent, one of them being hubness.

1.2 Problem Statement

Problem definitions from different papers include:

- Hubness is a phenomenon which occurs while analyzing high dimensional datasets. It is the result of an asymmetric relationship between a point and its neighbor inside the

dataset. This asymmetric relationship occurs if a data point has a small distance to a great number of other points. [3]

- Hubs are influential objects which frequently occur as neighbors to other points. Most other points rarely occur as neighbors and thereby having little influence on subsequent classification. Further potential errors in hub points can easily propagate and compromise many k-neighbor sets. [4]

- Hubs are data points which have a significantly higher than expected probability of appearing in the kNN-lists of other points. On the contrary, anti-hubs are data points for which the probability of appearing in the kNN-list of other points is significantly lower. [5]

- In the similar item recommendation scenario, hubness has a negative effect as hubs occur in many of the top N recommendations and thus reduce coverage and reachability, particularly of long-tail items. [6]

- Another different problem definition is that hubness is exploited in high dimensional user-response data. Thereby fake users are injected which are likely to become hubs, in order to gain maximum control over the output of the system and significantly influence the system. [7]

This means that if you search similar data records of two completely different points in a dataset with high hubness, it could occur that the same point is found. That point then would be a hubpoint according to the upper definitions. That behavior is problematic because you could find a similar point which could be very different.

1.3 Goal of the Thesis

The first expected outcome of this thesis is a method to compute the hubness of a dataset using different distance metrics. The results should provide a foundation to compare the measurements of hubness using different metrics. Also, the created methods should be usable on any numerical datasets. After the computation the hubness should be reduced. The reduction will be accomplished by analyzing, implementing and executing different approaches. The methods are as follows:

- Reduction through excluding the points causing the most hubness. According to tests with independent and identically distributed random variables [5], points causing the most hubness are only a tiny fraction of the data set. This reduction has a minor negative impact of the quality of the dataset due to excluding some points, but a large improvement through eliminating most of the hubness.

- Reduction through projection on a hypersphere of corresponding dimension would provide a hub free dataset. [5] The back draw of this method would be that the complete dataset would have to be modified. Also, the changes to the dataset must be verified, so that it still represents the same data as before.

- Reduction through application of a Principal Component Analysis which reduces the dimensions of a dataset. The result is representing the most important dimensions of the dataset. Additionally this approach enables the visualization of a dataset with highlighted hubpoints.

The performance of the reduction methods can be compared by measuring the hubness repeatedly. From this, the following research questions are derived:

- How much hubness does a dataset contain using different distance metrics? (L1, L2, Linf, cosine distance)
- Can the hubness of a dataset be reduced?
  - What is the effect on the quality of the retrieval system?
  - Are the results still meaningful?
- What is a hub point composed of in the dataset?
  - What are the most important dimensions which are responsible for hub points?
  - How does excluding those dimensions impact the quality of the retrieval system?

1.4 Structure of the Thesis

State of the Art

The chapter *State of the Art* provides theory about hubness in general, including explanations for where and why hubness occurs. Also, the steps that are necessary to calculate hubness are explained. This chapter also contains the definitions of the used metrics and the distance matrix. Finally the Matlab hubness Toolbox is explained which was extended to make the calculation of the desired results possible.

Reduction Methods

The chapter *Reduction Methods* first explains the extensions made to the Hubness Toolbox. Then it explains the concept and functionality of the three proposed reduction methods. This includes theoretical explanations and the core elements of matlab code. Lastly it is shown that all three reduction methods reduce hubness as proposed using artificial datasets.

Data

The chapter *Data* shows the used datasets and can be divided into three parts. The first part describes the different artificial datasets generated by matlab. It includes popular distributions, namely Standard Normaldistribution, Uniform Distribution and Exponential Distribution. The next part describes the “real world” classification sets which were used to test the quality of the retrieval system. Lastly the million song dataset is used to show the limits of computability.

Experiments

The chapter *Experiments* shows the computation work that was done. The results of the different ways to calculate the distance matrix on demand are compared. They make it possible to calculate the hubness of bigger datasets. Also, the initial hubness values and the reduced hubness values are compared. The last part focuses on the quality of the data and the retrieval system. There it is shown that on the one hand the PCA reduction method reduces hubness but also reduces the quality, and on the other hand the projection method reduces hubness and improves the quality slightly.

CHAPTER 2

State of the Art

2.1 Introduction

This chapter provides an introduction to the theoretical side of hubness. It defines metrics and lists the used metrics. Then it is explained how the hubness value is calculated using different distance metrics. Also the quantitative methods which were used to answer the research questions are explained. Finally, the chapter provides insight into the original hubness toolbox in Matlab that was used as foundation for all calculations.

2.2 Metrics

The following definitions are from the linear algebra book. [8] A metric characterizes the size or length of a vector. It is a transformation of a vector **V** into a real number.

$$\|.\| : \mathbf{V} \to \mathbb{R} : \mathbf{x} \to \|\mathbf{x}\| \tag{2.1}$$

It has the following properties:

1. $\|\mathbf{x}\| = 0$ follows $\mathbf{x} = 0$ for all $\mathbf{x} \in \mathbf{V}$
2. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ for all $\mathbf{x}, \mathbf{y} \in \mathbf{V}$
3. $\|c\mathbf{x}\| = |c| + \|\mathbf{x}\|$ for all $c \in K$ and $\mathbf{x} \in \mathbf{V}$

The definition of the p-Norm as foundation for the L1, L2 and Linf norm is as follows:

$$\|(x_1, x_2, ..., x_n)^T\|_p := \sqrt[p]{|x_1|^p + |x_2|^p + ... + |x_n|^p} \tag{2.2}$$

For p $\to \infty$ the norm is called uniform norm or Chebyshev norm:

$$\|(x_1, x_2, ..., x_n)^T\|_\infty := max(|x_1| + |x_2| + ... + |x_n|) \tag{2.3}$$

A metric space needs to have following properties:

1. dist(x,y) = dist(y,x) ≥ 0
2. dist(x,y) = 0 if x = y
3. dist(x,z) $\leq$ dist(x,y) + dist(y,z)

There exists a range of different metrics. Matlabs pdist2 function, which calculates the distance between two points, contains eleven different standard distance metrics. Yet in this thesis we will focus on the following widely-used metrics.

L1 norm

The L1 norm, also called Manhattan distance is the p-Norm with p=1. It describes the distance by the number of steps it takes from x to the other point y were only grid-like movement is allowed.

$$d_{l1}(x,y) = \sum_i |x_i - y_i| \tag{2.4}$$

L2 norm

The L2 norm or Euclidean distance is the p-Norm with p=2. It is the straight line distance between two points x and y.

$$d_{l2}(x,y) = \sqrt{\sum_i (x_i - y_i)^2} \tag{2.5}$$

Linf norm

The L infinity norm is also called Chebyshev norm and represents the p-Norm with p $\rightarrow \infty$.

$$d_{linf}(x,y) = max_i(|x_i - y_i|) \tag{2.6}$$

cosine norm

The cosine norm describes the angle between two points.

$$d_{cosine}(x,y) = \frac{\sum_i (x_i \cdot y_i)}{\|x\|_{l2} \cdot \|y\|_{l2}} \tag{2.7}$$

Using one of those metrics a distance matrix with $n \times n$ can be calculated. Each value d_{ij} represents the distance between point i and point j. The diagonal is equivalent to the distance to itself which is always 0.

$$\begin{bmatrix} d_{1,1} & d_{2,1} & \dots & d_{i,1} \\ d_{1,2} & d_{2,2} & \dots & \vdots \\ \vdots & \vdots & \ddots & d_{i,j} \\ d_{1,j} & \dots & d_{i,j} & d_{i,i} \end{bmatrix}$$

2.3 Hubness Theory

When calculating distances in high dimensional spaces, two phenomena emerge. The first one is known as distance concentration which means that pairwise distances between two points tend to become very similar. The second one, hubpoints, occur as a consequence. [6] Also, if the dimensionality d is sufficiently high, points lie almost exclusively at the vertices of a hypercube. In other words the distance structure of the hypercube vertices is equivalent to the distance structure of high dimensional points when d is sufficiently large. [5] A basic example [5] is a square like figure 2.1 (A) where all points are exactly at the vertices. Because of equal distances all points could be mutually nearest neighbors to each other, therefore low hubness. Now in the case of figure 2.1 (B) when one point is distorted to the mean chances that P1 is the nearest neighbor of points P2 and P4 and therefore making P1 a hub. In [9] it is stated that it can be

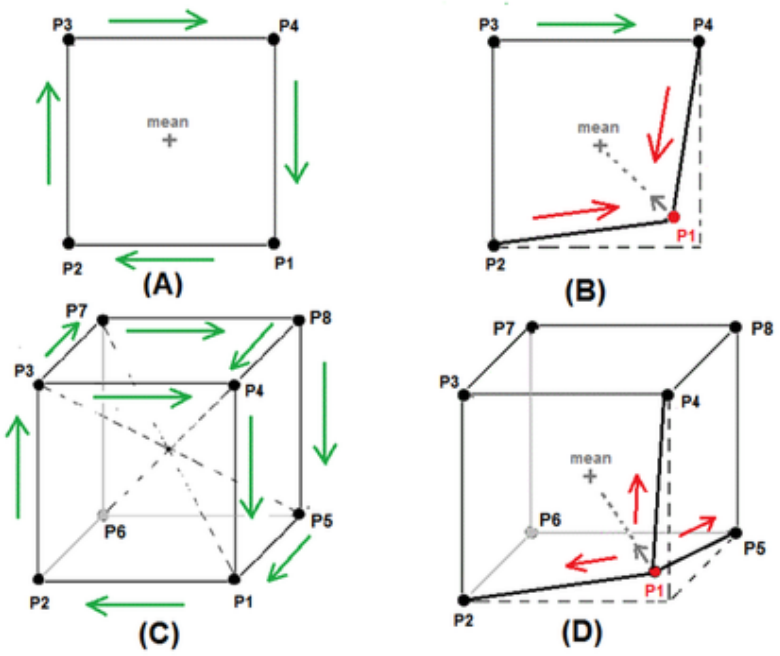


Figure 2.1: This figure from [5] illustrates the difference between symmetrical nearest neighbor relations in A with dimension 2 and in C with dimension 3 to the same example with one point distorted towards the mean. This changes the nearest neighbor relation so that the distorted point becomes a hubpoint.

expected in high dimensional datasets that there exists a non-negligible number of points closer to the dataset mean. Those points which are closer to the mean tend to be closer to all other points, which makes them hub points. On the other hand there are also points expected which are farther from the mean and also farther from all other points. They can be regarded as outliers.

Another simple example can be seen in Figure 2.2 showing a couple of points. It can be seen that the one point in the middle is often found as neighbor and the surrounding points are

rarely found as neighbors. Hubness occurs in high dimensional data set were some data points

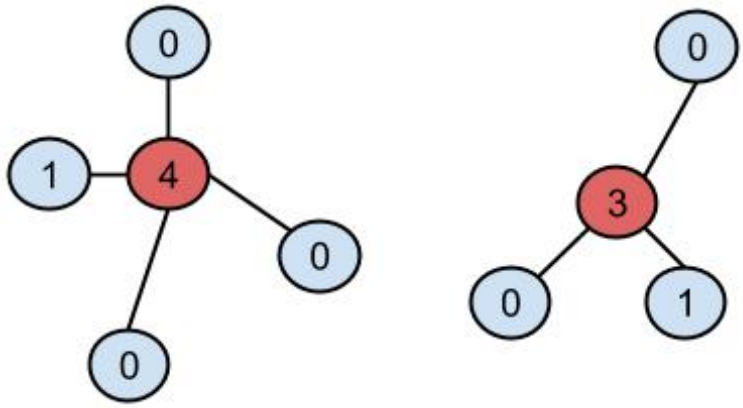


Figure 2.2: This figure illustrates the phenomenon of hubness in a simple two dimensional example. In this sample k is chosen to be 1 for simplicity which means that only one nearest neighbor is found. The red points are hub points and the number inside the point describes the n-occurrences value which is the number of times it is found in the k-nearest neighbor set of all other points.

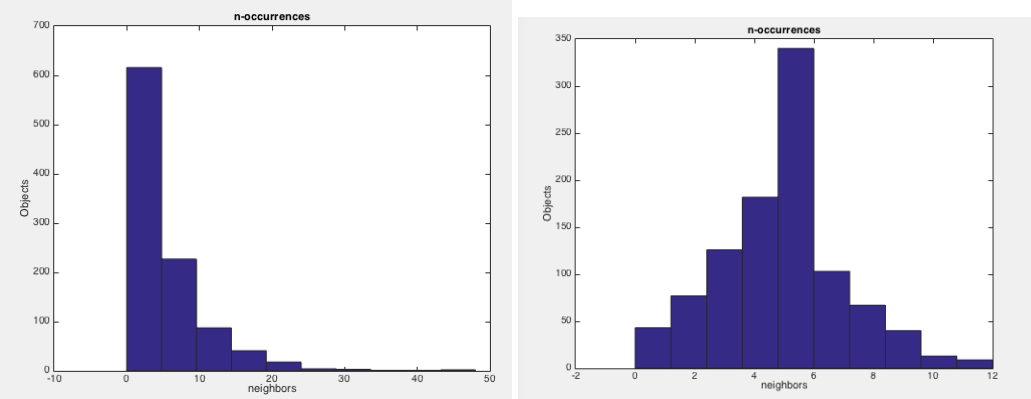
have small distances to many other data points, which effect many machine learning tasks. [3] Therefore the hub points are often found in the k-neighbor set and anti-hubs are rarely found. “Hubness is a geometric property of inherently high-dimensional data, as the points closer to the centers of hyper-spheres, where most of the data lies, tend to become very similar to many data points and are hence often included as neighbors.” [4]

It can be computed by calculating the n-occurrences of every point. N-occurrences O^n is a vector which contains the number of times the current point is in the k-nearest neighbors of all other points.(see Figure 2.3) Most points with no neighbors and some with many neighbors (see Figure 2.3a) indicates high hubness, whereas the opposite, many points with equal neighbors (see Figure 2.3b) indicates a low hubness. The skewness (third standardized moment) of this vector is defined as hubness, positive skewness indicating high hubness and on the contrary, negative skewness indicating low hubness. [3]

$$S^n = \frac{E[(O^n - \mu_{O^n})^3]}{\sigma^3_{O^n}} \tag{2.8}$$

Looking at two distributions, a normal distribution and a uniform distribution with the same variance, the normal distribution is more similar to the hubness-optimal distribution, because the majority of points is located around the mean and its tails flatten out. Also the normal distribution has a higher kurtosis and exzess (fourth standardized moment) which can be used as a hubness indicator. [5] The following definitions are taken from a statistic script. [10]

$$kurt(Y) = \frac{E[(Y - E[Y])^4]}{(E[(Y - E[Y])^2])^2} \tag{2.9}$$



(a) high hubness (b) low hubness

Figure 2.3: This figure illustrates the difference between a high and low hubness dataset. It shows two different distributions of n-occurrences. The high hubness distribution can be identified by looking at the high value with few neighbors and a few values with extraordinarily many neighbors. The low hubness distribution shows a more equal distribution with few points with no neighbors and also few with many neighbors.

$$Exzess = kurt(Y) - 3 \tag{2.10}$$

This function can be explained in words as the vector Y minus the expectation of Y which is the mean. Take that to the power of four and divide it through the variance to the power of two.

2.4 Quantitative methods

To answer the research questions two different methods were used. The first is a statistical test, the Wilcoxon test [11], which is used to answer the research questions, if one metric is better than another and if one reduction is better than another. It tests the null hypothesis if the medians of two samples are equal and is applicable even for a small sample number. It takes the paired score differences and ranks them in ascending order by absolute value. The sign of each difference is given to its rank as a label so that we will typically have a mix of “negative” and “positive” ranks.

Since I applied it one-sided, the hypothesis H0 proposes that the first median is less than or equal to the second median. The alternative hypothesis H1 is that the first median is bigger than the second median. This means in our tests, if the H0 hypothesis can be rejected, the method or metric is significantly better in terms of the produced hubness value. The Wilcoxon test is already implemented in matlab and can be used by executing following command:

```
[p,h,stats] =
ranksum(data(:,1),data(:,2),'tail','right','method','exact');
```

The second method which was used answer the research question regarding quality is a classification as porposed by [12]. Therefore a 10 fold cross validation was used to calculate the mean squared error as a performance value. A cross validation splits the dataset into a specified number V of blocks, iterates over each of the V blocks whereby each block is used once as test sample and calculating a error rate. Finally, the whole error rate is derived by averaging over all error results. This method is called V-fold cross validation. [13]

In our case the 10 folds were randomly assigned. The resulting values were used to test the meaning of the data and quality of the retrieval system. It was implemented as suggested in an example in the matlab documentation [14]:

```
cp = cvpartition(y,'k',10);
classf = @(XTRAIN, ytrain,XTEST)(classify(XTEST,XTRAIN,ytrain));
cvMCR = crossval('mcr',X,y,'predfun',classf,'partition',cp);
```

2.5 Hubness Toolbox

As foundation for the hubness calculations the "hub toolbox version 2" was used. It contains a collection of hub analysis tools and hub reduction tools. As a core part it contains a function to calculate the hubness value of a distance matrix. Additionally there is a method which calculates a distance matrix using the cosine metric. Also implemented are 3 reduction methods: local scaling, mutual proximity and shared nearest neighbor. [15] [12] Because the distance matrix is needed in the input, there is a need for a method that previously calculates the distance matrix. Additionally this missing function should contain the possibility to use the proposed distance metrics. Another problem that can arise is that the calculated distance matrix exceeds the memory by containing $n \times n$ double values. To solve that problem a method has to be implemented to calculate only the part of the distance matrix that is used in one iteration which is one line of the distance matrix.

The main part I used was the core of the "hubness" function. It handles the hubness calculation by iterating over each row of the distance matrix **D**, which has to be in the input. It then writes the first k neighbors into a new matrix **Dk**.

```
Dk = zeros(k, size(D, 2));
    for i = 1:size(D, 1)
        % extract distance matrix row
        d = D(i, :);

        % make non-finite (NaN, Inf) appear on the end of
        %the sorted list
        d(~isfinite(d)) = d_self;
        d(i) = d_self;

        % randomize the distance matrix row to avoid
        % the problem case if all numbers to sort are the
        % same, which would yield high
```

```
        % hubness, even if there is none.
        rp = randperm(size(D, 2));
        d2 = d(rp);
        [tmp, d2idx] = sort(d2, sortorder);
        Dk(:, i) = rp(d2idx(1:k));
    end
```

In the next step the n-occurrences vector Nk is calculated by searching for each point in the **Dk** matrix. The final hubness value is determined by calculating the skewness of the Nk vector.

```
    Nk = zeros(length(D), 1);
    for i = 1:length(D)
        Nk(i) = sum(sum(Dk == i));
    end

    Sn = local_skewness(Nk);
end

function s = local_skewness(x)
    x0 = x -  mean(x);
    s2 = mean(x0.^2);
    m3 = mean(x0.^3);
    s = m3 ./ s2.^(1.5);
end
```

2.6 Summary

Chapter two provides an explanation of the foundations of hubness. First, the metrics that were used during the experiment are introduced, defined and explained in detail. On a theoretical basis it is described why hubness occurs and how it can be measured. The reason for hubness is that some points are closer to the dataset mean and therefore are closer to all other points. To get the hubness value it is necessary to calculate the neighbors of each point as N-occurrences. The skewness of this vector is then defined as hubness. Furthermore the quantitative methods that were used to answer the research questions are explained. The last part of this chapter deals with explaining the unmodified hubness toolbox that was used to calculate the hubness of a dataset.

CHAPTER 3

Reduction Methods

3.1 Introduction

First this chapter explains the extensions of the different ways to compute the distance matrix and ways to deal with the problem of the limited memory. Further it covers the theory of the three hubness reduction methods proposed and used during the experiment. The first one being the exclusion of the biggest hub points. The second reduction method projects the datapoints onto a hypersphere, which was said to be almost hubfree. The third and final method reduces the dataset to the most important dimensions using principal component analysis. Explanation of the nature and implementation in matlab of each method are provided. Lastly, generated data is used to show how well the reductions work, differentiating the three methods and the four metrics.

3.2 Hubness Toolbox extensions

The first addition is the implementation of a method that calculates a distance matrix and is able to use the proposed metrics (L1, L2, Linf, cosine) since the Toolbox only accepts a distance matrix as input. The method pdist was used to accomplish that.

```
function D = distance(X,dist)
    switch dist
        case 'L1'
            D = pdist(X,'cityblock');
        case 'L2'
            Dtemp = pdist(X,'euclidean');
            D = squareform(Dtemp);
        case 'Linf'
            D = pdist(X,'chebychev');
        case 'cosine'
```

```
            Dtemp = pdist(X,'cosine');
            D = squareform(Dtemp);
        otherwise
            Dtemp = pdist(X,'euclidean');
            D = squareform(Dtemp);
    end
end
```

The problem that followed was the memory limit which was easily reached when calculating a distance matrix using the pdist function. It reaches the memory limit calculating a 30000×20 datamatrix which produces a 30000×30000 distance matrix full of double values. In terms of time it takes 2 seconds for a 10000×20 datamatrix, 15 seconds for a 20000×20 datamatrix and 75 seconds for a 30000×20 datamatrix. A possible solution for avoiding reaching the memory limit is to calculate the each part of the distance matrix exactly when and where it is needed. Therefore the part in the "hubness" function where one line of the distance matrix is extracted has to be replaced.

```
%d = D(i, :); %original; extract on row of the distance matrix

%on demand calculation of point i to all points
d = distance(D(i,:),D,dist);
```

Inside the distance function the distance is calculated using the "pdist2" instead of the "pdist" function both provided by matlab. This function provides the possibility to calculate the pairwise distance from one point p to the full dataset D. Additionally it also includes all proposed distance metrics which makes this function a perfect fit.

```
d = pdist2(p,D,'euclidean');
%calculates the distance from p to all points in D
```

This made the calculation a little slower (see Table 3.1) but solved the memory problem. The next optimizing modification which should improve the on demand-version was implementing a range search which doesn't calculate all neighbors for each row of the distance matrix. Calculating only the nearest k neighbors would be sufficient to fill the **Dk** matrix. This was implemented using the "rangesearch" function provided by matlab and trying to optimize the range value according to the quality of the result. Also the rangesearch function contains all proposed metrics.

```
[idx] = rangesearch(D,Dreidx,range,'Distance','euclidean');

%recieves how many neighbors were found for each point
    cellsz = cellfun(@size,idx,'uni',false);

    %how many points have more then k neighbors calculated
    reidx = ones(size(D,1),1);
    for i = 1:size(D, 1)
        if cellsz{i}(2)>k+1
```

```
            reidx(i)=0;
        end
    end

    if sum(reidx)<(size(D,1)/100*50) %if more than 50% is filled
        break;
    elseif sum(reidx)>(size(D,1)/100*80)%if less then 80% is filled
        range=range+40;
    elseif sum(reidx)>(size(D,1)/100*60)
        range=range+5;
    end
```

This method only calculates the distances to the points in range. If the number of points where at least k neighbors were found is too low, the radius is increased and the process is redone. This worked perfectly with regard to time efficiency but the initial memory problem reoccurred. Again it reaches the memory limit trying to calculate the hubness of a 30000×20 datamatrix. Modifying this approach to memory efficiency using the rangesearch for each row worked, but lost the time advantage to the initial on demand-version (see Table 3.1).

```
[idx] = rangesearch(Mdle,D(i,:),range,'Distance','euclidean');

%recieves how many neighbors were found for each point
        cellsz = cellfun(@size,idx,'uni',false);

        %if mor then k neighbors were found
        if cellsz{1}(2)>k+1
                Dk(:, i) = idx{1}(1:k+1);
        else
            i=i-1; %if not do again with adjusted range
        end

        %adjust range
        if cellsz{1}(2)<k+1
            range=range+5;
        elseif cellsz{1}(2)>(size(D,1)/100*10)
            range=range-10;
            if range<0
                range=1;
            end
        end
```

Another completely different approach was trying to implement the whole distance matrix into a memory mapped file during the creation. This solution did not work since calculating the whole distance like in the original-version using the “pdist” function could not be interrupted and using the “pdist2” function like in the on demand-version already solved the problem.

	Original	on demand	range	range on demand
1000×20	0,711843 sec	0,601333 sec	0,211714 sec	1,523194 sec
5000×20	7,725473 sec	10,294646 sec	1,260184 sec	9,364285 sec
10000×20	41,519911 sec	39,720651 sec	4,824376 sec	34,728445 sec
15000×20	164,810756 sec	73,309417 sec	11,256443 sec	72,043842 sec

Table 3.1: This table displays calculation times based on normal distributions with euclidean distance. Since the original version does not compute the distance matrix this has been added to the result time. The result times show that the original takes about as long as the on demand and the range on demand until the dataset is larger than 10000. The range method is by far the fastest and the two on demand methods are equally fast and generally similar to the original version.

Regarding time and memory as long as the dataset does not exceed a certain size depending on memory available (in my case 30000) the range method is clearly the fastest. For every dataset bigger than that I would recommend the normal on demand version because the range on demand version does not improve in time and both do not exceed the memory.

All calculations were made on a mac book with a 1,7 GHz Intel Core i5 processor and 4 GB 1333 MHz DDR3 ram.

3.3 Exclusion of hubpoints

The first reduction method is the reduction through excluding the points causing most hubness. According to tests with independent and identically distributed random variables, points causing the most hubness are only a tiny fraction of the data set. This means there is only a minor negative impact from excluding very few points through losing a tiny fraction of data points, but a large improvement through eliminating a significant amount of the hubness. In his work, Abdel Aziz Taha describes it as follows: “the hubness was reduced to 50% by removing 2% of the points on average” [5]. This reduction method can be accomplished by calculating the hubness once, taking the biggest values of the n-occurrences and removing those points from the matrix.

```
%sort the n-occurrence vector
[sortedX,sortingIndices] = sort(Nk5(:),'descend');

%create a index vetor with only 0
maxIndex = logical(zeros(length(D),1));

%change the index which are in the top 1% to 1
maxIndex(sortingIndices(1:((length(D)/100)-1)))=1;

%remove all rows of D were the index is 1
[Dremoved,PS] = removerows(D,'ind',maxIndex);
```

Applying this reduction method on a standard normal distributed dataset with $\mu = 0$ and $\sigma^2 = 1$(see Figure 4.1) it can be observed that each metric improves. The cosine metric stands out above the others in original hubness and reduced hubness (see Figure 4.1). Looking at a uniform distributed dataset (see Figure 4.2), and at an exponential distributed dataset (see Figure 4.3), an improvement is observable. The results improve even further if more than 1% of datapoints are removed.

3.4 Projection on a sphere

The reduction through projection onto a hypersphere would provide a hub free dataset. [5] This method projects the data points onto a sphere of correlating dimension. The downside of this method would be that the complete dataset would be modified. The changes to the dataset would need to be verified, so that the dataset still represents the same data as before. The projection method calculates the cartesian coordinates of an n-dimensional hypersphere. The n-hypersphere generalizes a circle for dimension n=2 or a sphere for dimension n=3. The n-hypersphere is therefore defined as the set of n-tuples of points $(x_1, x_2, ..., x_n)$ such that $x_1 2 + x_1^2 + ... + x_n^2 = r^2$. [16]

```
Dproj = HyperSphere([D],r);
```

The function's main component consists of a loop which could make this an impractical solution for large datasets.

```
for i = 1:N - 1
    y = [repmat(cos(X(i,:)),i,1)];
    y = [y ; sin(X(i,:))];
    y = [y ; ones(N - i - 1,n)];

    Y = Y.*y;
end
Y = Y*r;
```

After computing the letterrecognition dataset it became clear that the function was not behaving as expected. Matlab printed an error message saying that the cosine metric may not be appropriate. This could be seen in 3.1 where in the sphere block the cosine metric hubness value exceeds all other values. After inspection of the dataset it became visible that the main part of the dataset was zero or close to zero. Therefore the projection had to be redone. Eventually Stackoverflow [17] led to the solution which provided the correct result. The first line calculates the length of each point. Dividing each point through its length renders the length of each point exactly 1.

```
X=X'; %transpose matrix

%calclate length of each point
lx = repmat(sqrt(sum(X.^2,1)), [size(X,1) 1]);
```

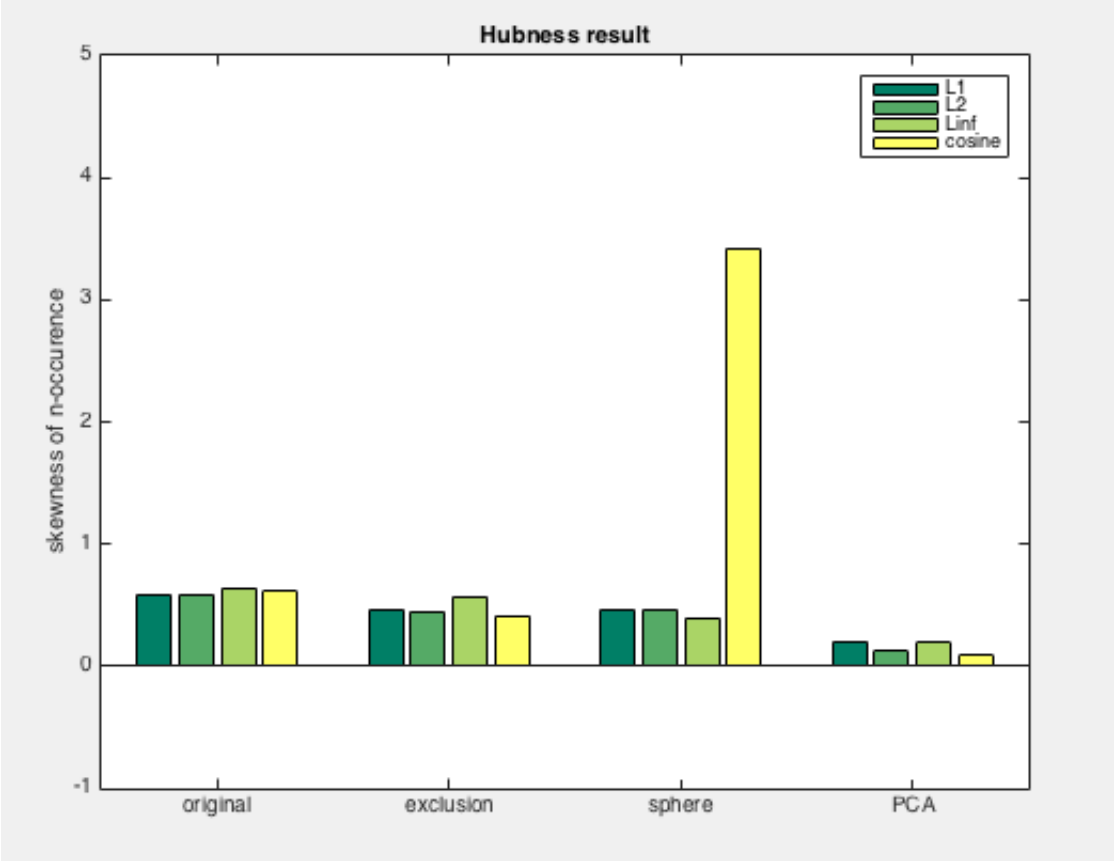


Figure 3.1: This image shows the initial result plot of the letter recognition dataset which made a misscalculation of the projection visible. The cosine metric produced many matlab warnings saying the cosine metric may not be appropriate. After inspection of the dataset it became visible that the main part of the dataset was zero or close to zero.

```
Y = X./lx; %divide each point by its length
Y=Y';
%for j = 1:size(X,2)
%norm(Y(j,:)) % Y(j,:) is the jth point on the circle)
%end
```

It can be verified that this procedure calculates the projection correctly since afterwards every point's length is exactly 1. The projection reduction method reduces the hubness on all artificially produced datasets.(see Figures 4.1, 4.2, 4.3) This seems to be a good reduction result, although it still needs to be shown that the meaning of the data did not get lost in the process.

3.5 Principal Component Analysis (PCA)

The following theory is from the data-analysis script which also inspired to use the principal component analysis for data-analysis purposes in general and as reduction method. [18] This reduction method, based on the measurements calculated, will identify the largest hubs and determine the reasons for their occurrences. Possibly the most important dimensions or a clustering habit can be detected. Also it would be possible that a dimension reduction could reduce the hubness of a dataset. A new datamatrix will be created where more than 80% of variance is described.

PCA in general is concerned with reducing dimensions of a dataset to make its structure visible while losing as little information as possible.

$$\mathbf{U} = \mathbf{X}\,\mathbf{B} \tag{3.1}$$

The data matrix **X** with the dimension $n \times p$ contains the original data records. **B** with the dimensions $p \times p$ contains coefficients to calculate the score Matrix **U**. The $u_1 \dots u_p$ of **U** are its Principal Components. The columns $b_1 \dots b_p$ of **B** are chosen so that each coefficient b_j produces the score vector u_j with maximum variance.

$$u_j = x_1 b_{1j} + \cdots + x_p b_{pj} \tag{3.2}$$

Furthermore b_j has to be orthogonal to previous loadings $b_j^T b_l = 0$ $(if j > l$ and $j >= 2)$ which maximizes the Variance of u_j.

$$Var(u_j) = b_j^T \Sigma b_j = b_j^T \lambda_j b_j = \lambda_j b_j^T b_j = \lambda_j \tag{3.3}$$

The b_j turn out to be the eigenvector of the covariance matrix.

```
[pc,score,latent,tsquare] = princomp(D);

%calculate the variance for each principal component
var = cumsum(latent)./sum(latent);
i = 3;
while var(i) < 0.8 %increasing i untill 80% variance is reached
    i = i + 1;
end
Dpca = score(:,1:i);
```

Trying the first approach on a matrix with the dimensions 1000×20 which is standard normal distributed with $\mu = 0$ and $\sigma^2 = 1$ it can be observed that the biggest hub points are not clustering.(see Figure 3.2) Using a uniform distributed matrix or an exponential distributed matrix with dimensions 1000×20 the results are similar. According to [4] "There is no easy way out, since dimensionality reduction techniques fail to eliminate the neighbor occurrence distribution skewness for any reasonable dimensionality of the projection space." This outcome seems plausible since the hubpoints are average points that are close to a lot of other points. The other appoach using PCA, the one creating a new datamatrix, showed better results. In most artificial datasets the hubness was reduced. (see Figures 4.1, 4.2, 4.3). If the meaning of the data is still the same, remains to be shown.

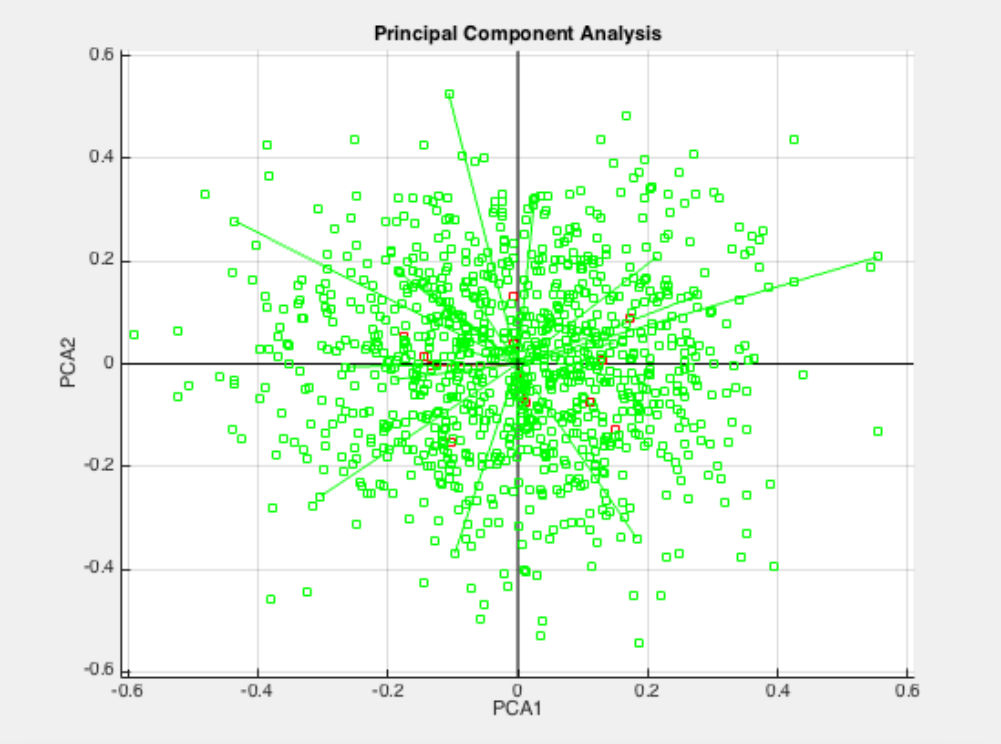


Figure 3.2: This figure shows one of many plots of the first two principal components. The red points show the hub points which are spread evenly around the plot. No clustering can be determined.

3.6 Summary

Chapter three deals with the modifications made on the hubness toolbox that was used to calculate the results. The most important one is the on demand calculation of the distance vector during the hubness calculation. Also other ways to compute the distance matrix and dealing with the problem of the limited memory are explained.

Further it covers the theory of the three hubness reduction methods proposed and used during the experiment. The exclusion of the biggest hub points results in a minor improvement. This comes naturally since the removal of the biggest hub points means less clustering and improved reachability of the surrounding points. The second reduction method projects the datapoints onto a hypersphere, which was said to be almost hubfree. The conducted calculations suggest that this theory is plausible. The results were not hubfree but still reducing the hubness of each dataset. PCA achieved also good results improving the original version.

Unfortunately, later on, during the comparison as part of the retrieval system experiment (Chapter 5.5) the PCA reduction method produced information loss of the data and thus made it unusable.

CHAPTER 4

Data

4.1 Introduction

The datasets described in this chapter can be segmented into generated datasets, classification datasets and the million song dataset. The generated datasets were used to implement the reduction methods and show that the methods produce the desired result. The classification datasets were used to test the quality of the retrieval system and show if the approaches are feasible. This approach tries to detect if the dataset was damaged in the process. Lastly the million song dataset, being 50 times as big as the letter recognition dataset, is used to show the limits of computability.

4.2 Generated Data

The datasets are artificially produced and randomly generated using matlab to illustrate different hubness values and tendencies for different distributions.

Standard Normaldistribution

The first artificially produced dataset is a standard normal distribution with $\mu = 0$ and $\sigma^2 = 1$. It has 10000 records and 20 dimensions. To create this table in matlab following command is used:

```
X=randn(10000,20);
```

The first four bars show the original hubness values of the dataset. It contains a high hubness value around 2 for L1, L2 and Linf. Measuring the hubness value with the cosine metric already returns good results at 0,3711. The second four bars show the hubness values after applying the exclusion reduction. After the biggest hubpoints were removed all metrics contained slightly less hubness. The hubness values of the L1, L2 and Linf metric improved but are still high

Hubness of a standard normal distribution

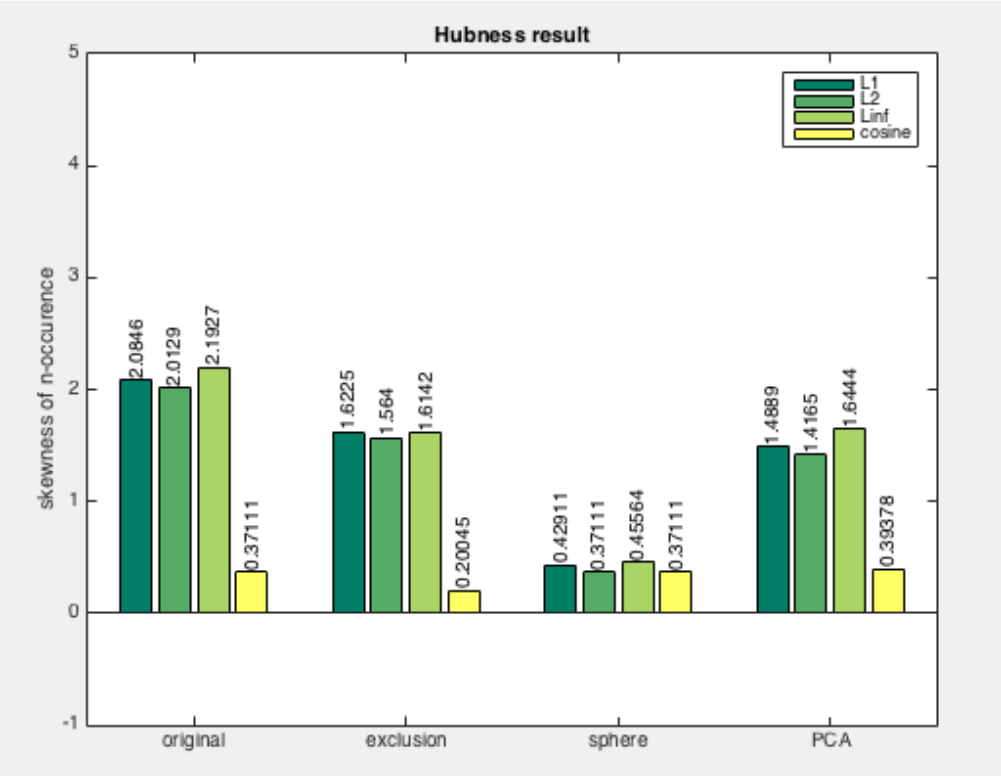


Figure 4.1: This image shows a plot of the contained hubness of a normal distribution. The plot shows initial high hubness. After the exclusion reduction and the PCA reduction the measured hubness is reduced slightly, the projection reduction reduced the measured hubness by a lot.

at around 1,6 and the cosine metric also improved and is still low at 0,2. The third four bars depict the contained hubness after executing the projection reduction. It can be observed that the initially high hubness values improve to a similar level as the cosine metric all around 0,4. The fourth four bars show the measured hubness after application of the PCA reduction which produced similar results as the exclusion reduction. Looking at this standard normal distribution the best options for producing a distance matrix with a low hubness value is either to choose the cosine metric or project all data points onto a hypersphere.

Uniform distribution

The second artificially produced dataset is an uniform distribution with random numbers equally distributed between 0 and 1 with 10000 records and 20 dimensions. To create this table in matlab following command is used:

```
X=rand(10000,20);
```

Hubness of a uniform distribution

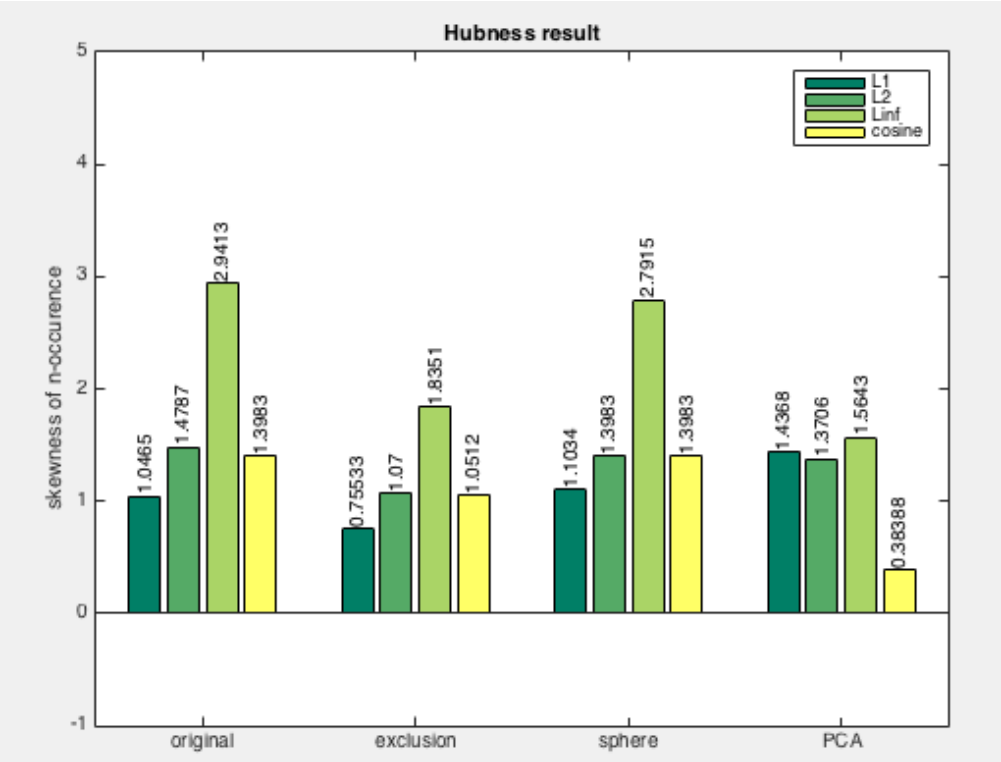


Figure 4.2: This image shows a plot of the contained hubness of an uniform distribution. The plot shows initial high hubness only for the Linf metric. After the exclusion reduction and projection on a sphere the measured hubness is reduced slightly, the PCA improved even a little more.

The hubness result plot of the uniform distribution shows initial medium hubness values slightly over 1 for the metrics L1, L2 and the cosine metric and a high value at 3 using the Linf metric. After application of the exclusion reduction all metrics show slightly improved hubness values 1 for L1, L2 and the cosine metric and still a high hubness value using the Linf metric. The third four bars show the results after application of the projection reduction which in this case produced about equal results as the initial values. The fourth four bars depict the contained hubness after the PCA reduction which produced medium hubness values at around 1,4 and one low value at 0,3838 using the cosine metric. In case of a uniform distribution the combination with the lowest hubness value is the csine metric in combination with the application of the PCA reduction. Unfortunately during the retrieval system comparison (Chapter 5.5) it turns out that the PCA reduction method produces a dataset with different meaning and thus made it unusable. Therefore the best combination in that case would be the L1 metric.

Exponential distribution

The third artificially produced dataset is an exponential distribution with mean parameter 2, 10000 datarecords and 20 dimensions. To create this table in matlab following command is used:

```
X = exprnd(2,10000,20);
```

Hubness of a exponential distribution

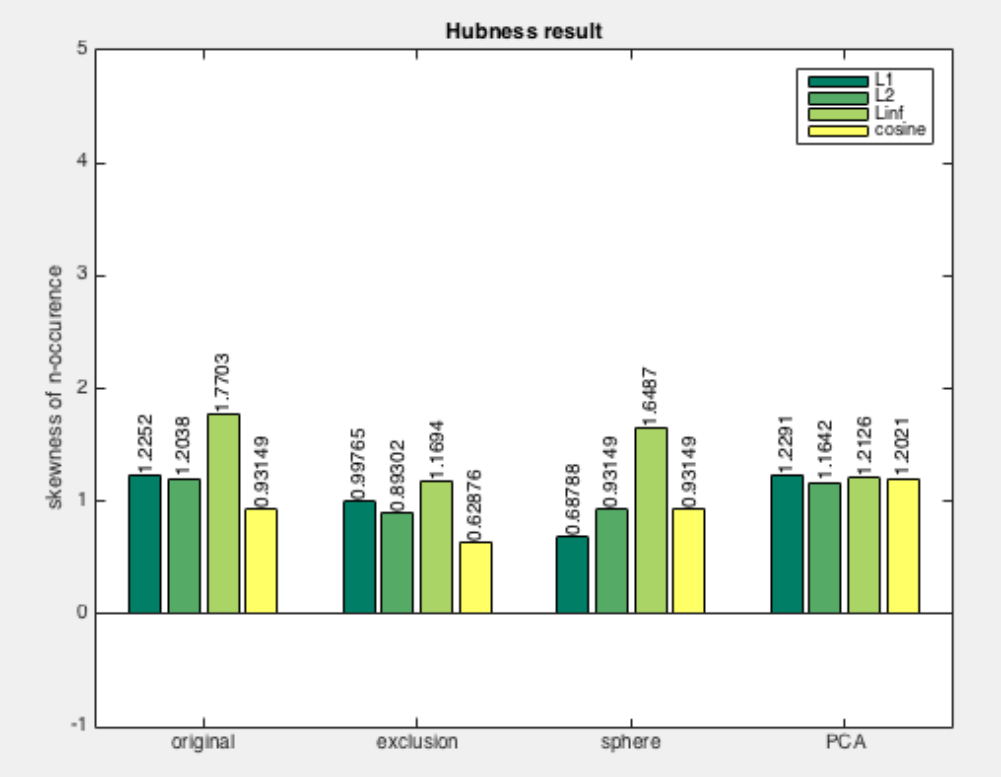


Figure 4.3: This image shows the plot of the contained hubness of an exponential distribution. The plot shows initial high hubness only for the Linf metric. After the exclusion reduction and projection on a sphere the measured hubness is reduced slightly, the PCA reduction did not effect much.

The hubness result plot of the exponential distribution shows a medium initial hubness of about 1 and again Linf slightly higher and the cosine metric slightly lower. The exclusion reduction improved all results a little where the cosine metric still contains the lowest hubness. Similar to the uniform distribution the sphere produced similar values as the initial hubness. The PCA reduction produced a medium hubness value of 1,2 for all metrics. The metric which contains the lowest hubness in this case is the cosine metric.

4.3 Wine quality

The wine quality dataset contains wine quality samples from May 2004 to February 2007 of "vinho verde", a unique summer wine from Portugal. It is distinguished in red and white since the taste is quite different. The table consists of the most common physicochemical tests which are: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates and alcohol. The class was determined by assessors using blind tastes who rated from 0 (very bad) to 10 (excellent) and is named quality. Since the quality values 0, 1, 2, 9 and 10 were never assigned the dataset remains with 6 classes. It has 1599 red wine instances and 4898 white wine instances. [19]

4.4 Letter recognition

The letter recognition dataset consists of a large number of black-and-white rectangular pixels which display an upper-case letter in 20 different fonts. The set has 20000 instances which are evenly distributed from "A" to "Z". The dataset has 16 dimensions which are all integer. They are statistical moments, edge counts, pixel counts and boxposition which were scaled to fit into integer values from 1 through 15. [20]

4.5 Million Song Dataset

The Million Song Dataset is a free collection of audio metadata and is proposed to be used as a large-scale "real world" dataset. It contains 1.000.000 songs/files, 44.745 unique artists, 2.201.916 asymmetric similarity relationships and 515.576 dated tracks starting from 1922. The main acoustic features are pitches, timbre and loudness for every segment. The dataset's limitations are missing album metadata, song-level metadata and tags. [21] I used a compressed table containing 8 dimensions and 994.623 instances. The 9th dimension is the TrackID.

4.6 Summary

Chapter four explained the creation of the generated datasets and the origin of the "real-world" datasets. The artificial datasets consist of random numbers which are normal distributed, standard normal distributed and exponential distributed. While the main focus of the experiment was the computing of different "real-world" datasets, the randomly generated numbers were mainly used to implement the reduction methods and to test the validity of the results.

CHAPTER 5

Experiments

5.1 Introduction

The experiments in this chapter aim to compare different hubness values in different datasets using different metrics. The first subsection shows a comparison of the calculated distance matrix to the estimated one that was achieved using a rangesearch. The next subsection deals with the comparison of hubness values, specifically those of the "real-world" datasets. All results are depicted in plots and explained in form of text. Additionally a Wilcoxon rank sum test was executed to show statistical significance. The "Hubness reduction" subsection shows the results obtained by executing the three reduction methods. The last subsection compares the retrieval system before and after each reduction. This was achieved comparing the missclassification rates by using a 10 fold cross validation.

5.2 Comparison of estimated and calculated hubness

The calculation of hubness takes a lot of time since the neighbors of n points have to be calculated n times. This process can be sped up by implementing a rangesearch for the neighbor search to calculate the distance matrix. This distance matrix that is required has the dimensions $n \times k$ and contains only the k nearest neighbor whereby k can be chosen. The range value is constant and gets adjusted according to the sparsity of the distance matrix. It is sufficient if the majority of the matrix is calculated since the points with no neighbors within the range are no hubpoints anyway according to the hubness definition.(see Table 5.4) When comparing tables 5.1 - 5.4 it can be seen that all methods provide the same Dk result except the range on demand version which leaves some points who have no close neighbors 0.

To compare the different hubness values of the different methods a bar plot was plotted and the different methods where plotted in different colors.(see Figure 5.1) There the green bars serve as the correct values computed with the on demand version of the original hubness toolbox. It

can be observed that the red bars, which show the results of the range search, show almost the same results. The blue bars which show the range ondemand search deviate a little more.

In combination with the time and memory results of the extensions from table 3.1) in the previous chapter it can be concluded that the rangesearch outperforms the other methods in in the time aspect while providing the same result. However it is still limited to memory which makes it unusable for datasets with more then 30000 datarecords. In that case a on demand method has to be selected. Since both take equally long and are equally memory independent the original on demand method is the better method. There is no benefit in accepting slightly inaccurate result hubness values which would be provided from the range search ondemand version.

5.3 Comparison of hubness using different distance metrics

The wine quality dataset initially contains very low hubness. This can be observed from the resultplots of both datasets (red and white) (see Figure 5.2a and 5.2b). There is also little difference between the different distance metrics. Like in the winequality dataset the letter recognition dataset has low initial hubness and equally low hubness for each metric used.(see Figure 5.3b) The hubness computation of the million song dataset showed similar results until the system crashed after a week of computation. One computation iteration took about 24 hours. The intermediate result described a initial hubness of 0.19 using the L1 metric. It is probably equally low for the other metrics, as every other non artificial dataset would suggest.

To analyze statistical differences of the distance metrics, a right sided Wilcoxon rank sum test was executed testing the metrics against each other. The null hypothesis (H0) of this test states that the median $\tilde{x}$ and $\tilde{y}$ are equal. The alternative (H1) states that the median $\tilde{x}$ is bigger than $\tilde{y}$ which means for our case that y produces a smaller hubness value. For the artificial datasets, H0 can be discarded for the Linf and cosine metric, meaning that the cosine metric has a lower hubness value than the other metrics and Linf has a higher hubness value than the other metrics.(see Table 5.5 5.6) For the non artificial datasets, H0 can never be discarded, which means that no metric is better than another one.(see Table 5.7 5.8)

5.4 Hubness reduction

Since the initial hubness of the wine quality dataset was very low, no reduction method improved the result visually (see Figure 5.2a and 5.2b). It is similar for the letterrecoginition dataset (see Figure 5.3b). The intermediate result of analyzing the million song dataset also showed little improvement because the initial hubness also was low. Exclusion reduced from 0.19 to 0.04 and PCA remained the same at 0.19 for the metric L1.

As already done in chapter 5.3 a right sided Wilcoxon rank sum test was executed testing the statistical difference between the reduction methods. As above, and in general, the null hypothesis (H0) of this test states that $\tilde{x}$ and $\tilde{y}$ are equal. The alternative (H1) states that $\tilde{x}$ is bigger than $\tilde{y}$ which again would mean that y produces a lower hubness value and is therefore better. The artificial datasets resulted in discarding H0 for the original hubness value compared to exclusion reduction and projection reductions. That means that those reductions render better

results than the original. (see Table 5.9 5.10) Different results can be seen in the calculations of the non artificial datasets. The exclusion and PCA reduction improve the original hubness value. They also result in lower hubness values than the projection reduction.(see Table 5.11 5.12)

5.5 Comparison of retrieval system before and after hubness reduction

Since often no ground truth or similarities are known, a classification was executed. This tries to show that the classification result improves after the hubness reduction has been executed. Therefore a 10 fold cross validation was executed, which divides the dataset into ten parts. Nine of them are used as training data and the remaining one is used as test data. After 10 iterations it lead to the following misclassification rates. (see Table 5.13) The misclassification rate represents the percentage of data records which were assigned to the wrong class. This shows that if the biggest hubpoints are removed, the quality of the retrieval system stays similar. A PCA-dimensionreduction will have huge negative impacts and damage the data meaning. The projection function will not only reduces the hubness of a dataset, but also slightly increases the quality of the retrieval system.

5.6 Summary

Chapter five depicts the actual computing work that was done during the experiment and its results. It can be observed that the on demand calculation has the similar result as the rangesearch. Since the rangesearch method turned out to be neither more time nor memory efficient during the initial trials, all calculations were executed using the on demand calculations. The original hubness values of the “real-world” datasets were generally low, which is why the reduction algorithms visually didn’t show as much effectiveness as in the generated data. The statistical analysis however revealed that exclusion and PCA improve the result significantly. Regarding the quality of the retrieval system, PCA reduction worsened the result. The projection on the hypershere on the other hand was the only method which improved the quality. The exclusion had neither positive nor negative impact on the quality.

Original	**Point 1**	**Point 2**	**Point 3**	**Point 4**	**Point 5**
neighbor 1	671	466	216	76	441
neighbor 2	391	278	239	810	589
neighbor 3	221	994	954	30	920
neighbor 4	623	29	503	333	50
neighbor 5	638	421	980	529	909

Table 5.1: This table displays the first 5 points of the distance matrix from calculating the hubness of a 1000×20 normal distribution using the original version of the hubness toolbox. It resulted in a hubness value of 3,0509.

Dk on demand	**Point 1**	**Point 2**	**Point 3**	**Point 4**	**Point 5**
neighbor 1	671	466	216	76	441
neighbor 2	391	278	239	810	589
neighbor 3	221	994	954	30	920
neighbor 4	623	29	503	333	50
neighbor 5	638	421	980	529	909

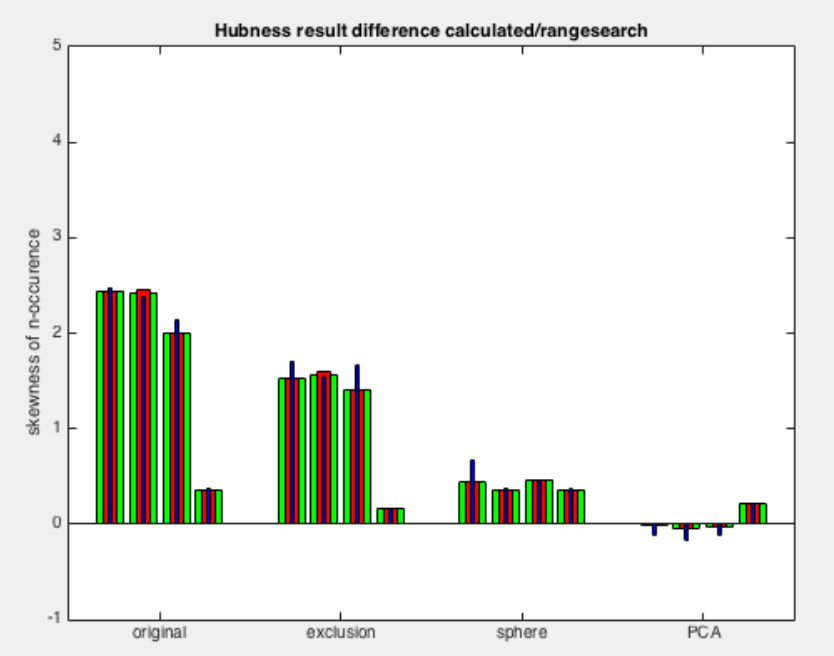
Table 5.2: This table displays the first 5 points of the distance matrix from calculating the hubness of the same 1000×20 normal distribution as in the original version using the adjusted on demand version. It resulted in the same distance matrix and the same hubness value as in the original version.

Dk range	**Point 1**	**Point 2**	**Point 3**	**Point 4**	**Point 5**
neighbor 1	671	466	216	76	441
neighbor 2	391	278	239	810	589
neighbor 3	221	994	954	30	920
neighbor 4	623	29	503	333	50
neighbor 5	638	421	980	529	909

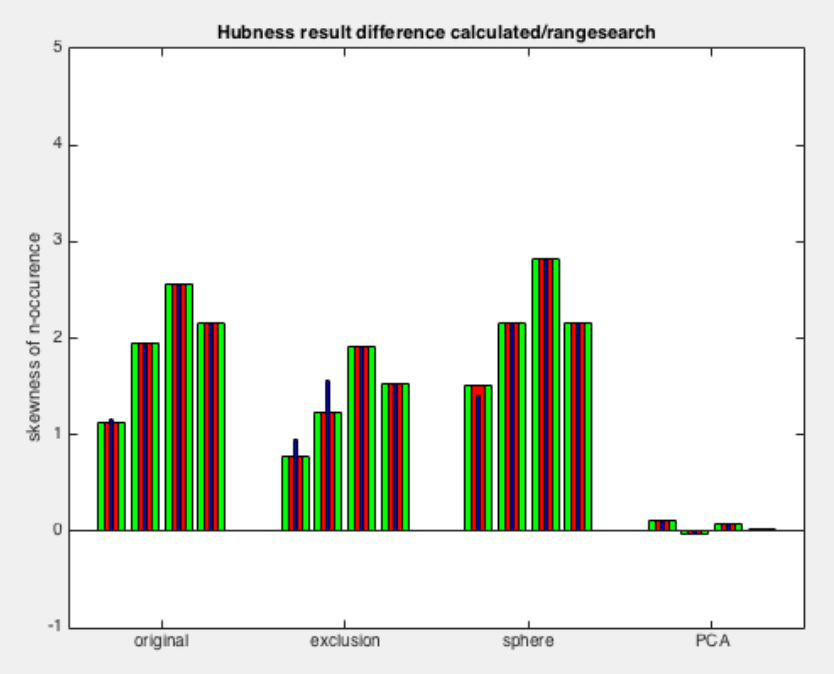
Table 5.3: This table displays the first 5 points of the distance matrix from calculating the hubness of the same 1000×20 normal distribution as in the original version using the adjusted range search version. It resulted in the same distance matrix and the same hubness value as in the original version.

Dk range on demand	**Point 1**	**Point 2**	**Point 3**	**Point 4**	**Point 5**
neighbor 1	0	466	0	76	0
neighbor 2	0	278	0	810	0
neighbor 3	0	994	0	30	0
neighbor 4	0	29	0	333	0
neighbor 5	0	421	0	529	0

Table 5.4: This table displays the first 5 points of the distance matrix from calculating the hubness of the same 1000×20 normal distribution as in the original version using the adjusted range search on demand version. It resulted in a similar distance matrix for all points which are near and 0 for all points which are far away. These do not influence the resulting hubness value since those are no hubpoints according to the definition. The resulting hubness value is 2.9675 which is almost the same as in the original.

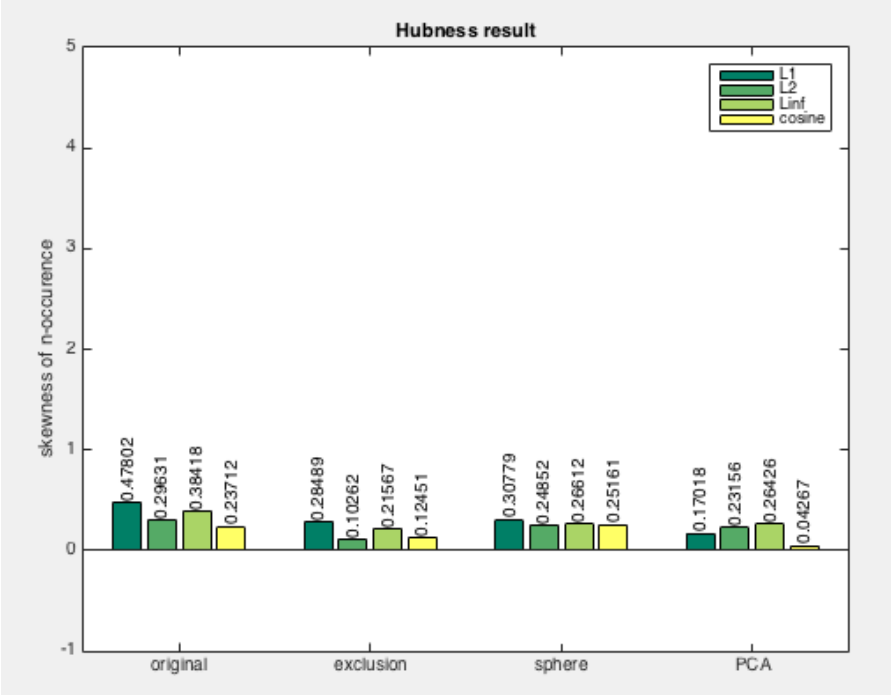


(a) normaldistribution 1000x20

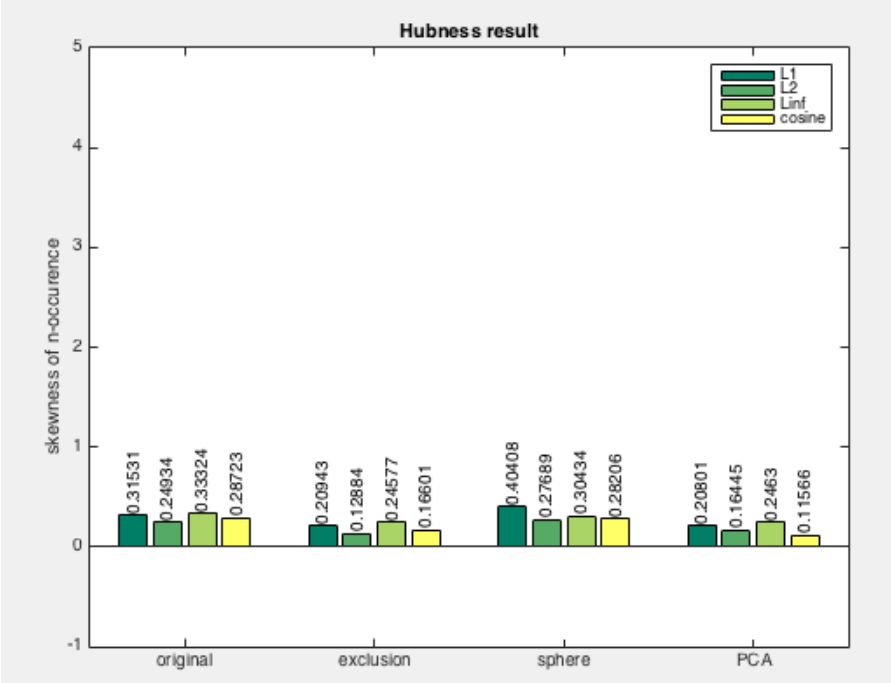


(b) uniformdistribution 1000x20

Figure 5.1: This plot compares the on demand results to the range results and the range on demand results. The original version is plotted green, the red bars depicts the range search results and the blue are the range on demand results. The range results are almost identical to the original, the range on demand has some deviation

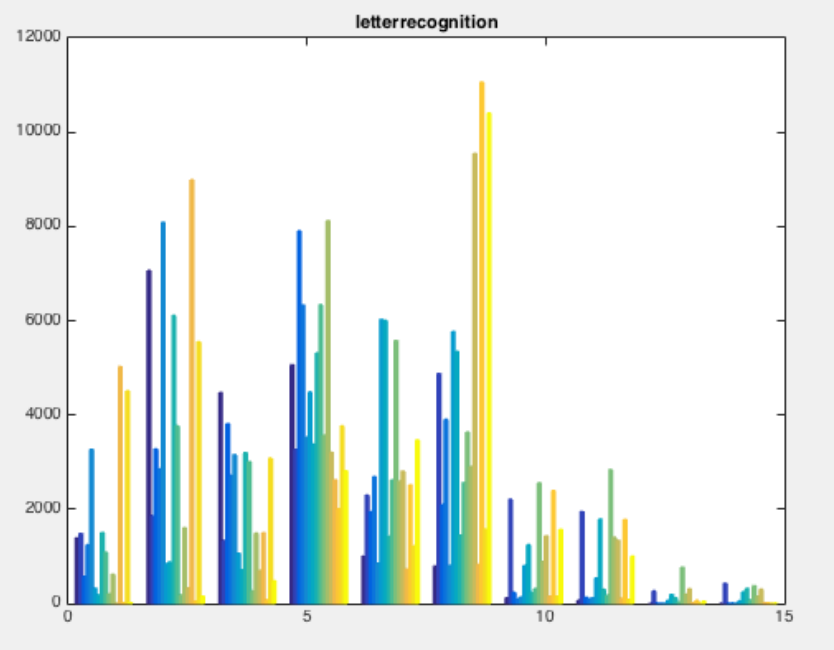


(a) result wine quality red

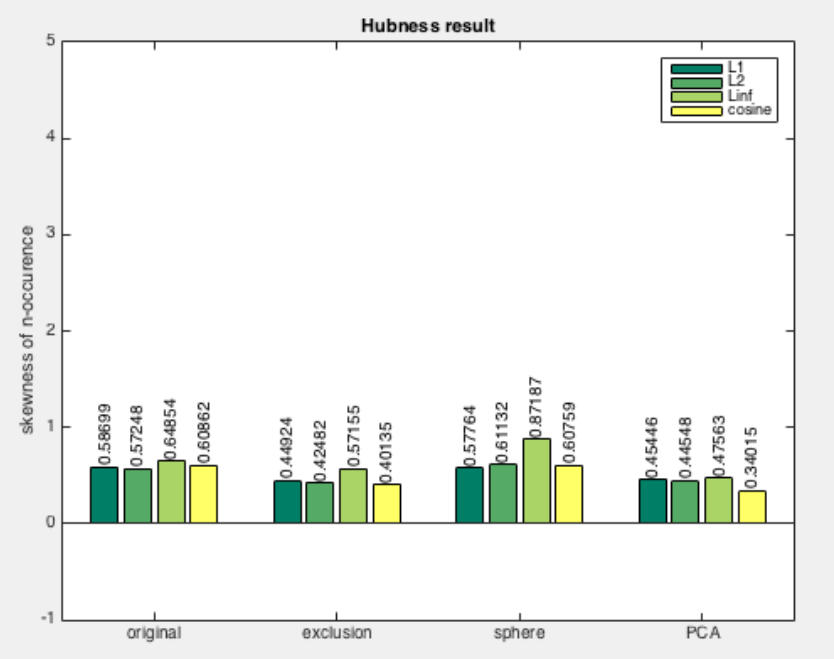


(b) result wine quality white

Figure 5.2: This plot is the first analysis executed on a non artificial dataset. It can be observed that the initial hubness is low which also makes all reduction methods ineffective. The y-axis is fixed to make each bar comparable.



(a) distribution of the letter recognition dataset



(b) result letterrecognition

Figure 5.3: This plot shows another non artificial dataset. Like before the initial hubness is very low which again makes the reduction methods ineffective. Again the y-axis is fixed to make each bar comparable.

Wilcoxon rank sum test results of the artificial datasets comparing metrics

p values	L1	L2	Linf	cosine
L1	0.5227	0.6647	0.9940	0.0186
L2	0.3563	0.5227	0.9914	0.0222
Linf	0.0072	0.0102	0.5227	0.0001
cosine	0.9828	0.9809	0.9999	0.5155

Table 5.5: This table describes the p value results from the one sided Wilcoxon rank sum test applied on the result hubness values from the artificial datasets. It focuses on the difference between the metrics.

h values	L1	L2	Linf	cosine
L1	H0	H0	H0	H1
L2	H0	H0	H0	H1
Linf	H1	H1	H0	H1
cosine	H0	H0	H0	H0

Table 5.6: This table shows the chosen hypothesis from the one sided Wilcoxon rank sum test applied on the result hubness values from the artificial datasets focusing on the difference between the metrics. H0 states that the hypothesis H0 can not be discarded.

Wilcoxon rank sum test results of the non artificial datasets comparing metrics

p values	L1	L2	Linf	cosine
L1	0.5227	0.1328	0.5786	0.1095
L2	0.8792	0.5227	0.8792	0.4278
Linf	0.4437	0.1328	0.5227	0.0989
cosine	0.9011	0.5849	0.9109	0.5227

Table 5.7: This table describes the p value results from the one sided Wilcoxon rank sum test applied on the result hubness values from the non artificial datasets. It focuses on the difference between the metrics.

h values	L1	L2	Linf	cosine
L1	H0	H0	H0	H0
L2	H0	H0	H0	H0
Linf	H0	H0	H0	H0
cosine	H0	H0	H0	H0

Table 5.8: This table shows the chosen hypothesis from the one sided Wilcoxon rank sum test applied on the result hubness values from the non artificial datasets focusing on the difference between the metrics.. H0 states that the hypothesis H0 can not be discarded.

Wilcoxon rank sum test results of the artificial datasets comparing reduction methods

p values	original	exclusion	projection	PCA
original	0.5227	0.0444	0.0386	0.1328
exclusion	0.9610	0.5227	0.2707	0.7952
projection	0.9662	0.7387	0.5127	0.8511
PCA	0.8792	0.2214	0.1556	0.5227

Table 5.9: This table describes the p value results from the one sided Wilcoxon rank sum test applied on the result hubness values from the artificial datasets. It focuses on the difference between the reduction methods.

h values	original	exclusion	projection	PCA
original	H0	H1	H1	H0
exclusion	H0	H0	H0	H0
projection	H0	H0	H0	H0
PCA	H0	H0	H0	H0

Table 5.10: This table shows the chosen hypothesis from the one sided Wilcoxon rank sum test applied on the result hubness values from the artificial datasets focusing on the difference between the reduction methods. H0 states that the hypothesis H0 can not be discarded.

Wilcoxon rank sum test results of the non artificial datasets comparing reduction methods

p values	original	exclusion	projection	PCA
original	0.5227	0.0121	0.3505	0.0072
exclusion	0.9898	0.5227	0.9770	0.4887
projection	0.6590	0.0252	0.5119	0.0118
PCA	0.9940	0.5338	0.9895	0.5227

Table 5.11: This table describes the p value results from the one sided Wilcoxon rank sum test applied on the result hubness values from the non artificial datasets. It focuses on the difference between the reduction methods.

h values	original	exclusion	projection	PCA
original	H0	H1	H0	H1
exclusion	H0	H0	H0	H0
projection	H0	H1	H0	H1
PCA	H0	H0	H0	H0

Table 5.12: This table shows the chosen hypothesis from the one sided Wilcoxon rank sum test applied on the result hubness values from the non artificial datasets focusing on the difference between the reduction methods. H0 states that the hypothesis H0 can not be discarded.

	Original	**removed**	**pca**	**projection**	**projection toolbox**	**random**
wine quality red	0.5647	0.56	0.7223	0.5291	0.8956	0.83
wine quality white	0.6874	0.68	0.8636	0.6345	0.8810	0.6370
letterrecognition	0.2984	0.29	0.4646	0.2976	0.9613	0.9375

Table 5.13: This table displays the misclassification rates using a 10 fold cross validation. It shows that the PCA reduction and the first version of the projection have a negative impact on the data. The exclusion reduction doesn't influence the data quality while the projection even improves it slightly.

CHAPTER 6

Conclusion

The aim of this thesis was to create three methods able of reducing the hubness of a dataset. This goal was accomplished. The attempt to improve the quality of the retrieval system failed for the PCA reduction due to the nature of the reduction method. The exclusion reduction neither improved nor reduced the quality of the retrieval system. The projection achieved both goals, reducing hubness and improving the quality of the retrieval system. The experiment also revealed that the initial hubness of "real-world" datasets seems to be generally lower, as all three of the samples would suggest. The accumulated data would suggest following answers to the research questions:

How much hubness does a dataset contain using different distance metrics? (L1, L2, Linf, cosine distance)

The extensions to the hubness toolbox make it possible to calculate the hubness for any dataset. When focusing on the "real world" datasets, all metrics seem to result in a similar hubness value. When focusing on the artificial datasets, the only metric that is statistically significantly better is the cosine metric and the only one which is significantly worse is the Linf metric. In the other cases the null hypothesis can not be discarded, which means that they can't be differentiated.

Can the hubness of a dataset be reduced?

All three proposed reduction methods reduced the hubness. Visually all reductions achieved a minor reduction. The statistical tests show that every reduction method except the pca reduction on the artificial data and the projection on the non artificial datasets are able to reject the null hypothesis in comparison to the original hubness values.

What is the effect on the quality of the retrieval system?

The results from the 10 fold cross validation show, that the exclusion reduction did not influence

the quality of the retrieval system. The result of the pca reduction showed a reduced quality, which made it practically useless. Projecting the data onto a hypersphere provided a slight improvement of the retrieval system.

Are the results still meaningful?

As stated in the previous answer, the pca reduction damaged the meaning of the dataset. The other two reduction methods did not change the meaning of the dataset.

What is a hub point composed of in the dataset?

To answer this question I visualized the dataset using PCA and colored the hubpoints in a different color. I had hoped to see a clustering habit, but that was not the case. Therefore the most important dimensions could not be localized and they could not be distinguished from antihub points.

What are the most important dimensions which are responsible for hub points?

Since the first approach, visualizing the dataset and hubpoints, did not show the expected results, this question inspired the PCA reduction method. It created a new data matrix using a linear combination which represented the most important dimensions, containing more than 80% of variance. This rendered slightly improved results in terms of reduction, but destroyed the quality of the data and the retrieval system. The damaged quality made this method not viable.

How does excluding those dimensions impact the quality of the retrieval system?

As stated in the previous answer, the damaged quality of the data and the retrieval system made this method unusable.

CHAPTER 7

Appendix

Bibliography

[1] A. J. Hey, S. Tansley, K. M. Tolle, *et al.*, *The fourth paradigm: data-intensive scientific discovery*, vol. 1. Microsoft Research Redmond, WA, 2009.

[2] K. Patel, "Lowering the barrier to applying machine learning," in *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, (New York, NY, USA), pp. 2907–2910, ACM, 2010.

[3] A. Flexer and D. Schnitzer, "Can shared nearest neighbors reduce hubness in high-dimensional spaces?," in *Data Mining Workshops (ICDMW), 2013 IEEE 13th International Conference on*, pp. 460–467, IEEE, 2013.

[4] N. Tomašev and D. Mladenić, "Hubness-aware shared neighbor distances for high-dimensional k-nearest neighbor classification," *Knowledge and information systems*, vol. 39, no. 1, pp. 89–122, 2014.

[5] A. A. Taha, A. Hanbury, and A. Rauber, "Formal analysis of hubness, indicator and mitigation strategies," 2015.

[6] P. Knees, D. Schnitzer, and A. Flexer, "Improving neighborhood-based collaborative filtering by reducing hubness," in *Proceedings of International Conference on Multimedia Retrieval*, ICMR '14, (New York, NY, USA), pp. 161:161–161:168, ACM, 2014.

[7] K. Hara, I. Suzuki, K. Kobayashi, and K. Fukumizu, "Reducing hubness: A cause of vulnerability in recommender systems," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, (New York, NY, USA), pp. 815–818, ACM, 2015.

[8] H. Havlicek, *Lineare Algebra fuer Technische Mathematiker*. Heldermann Verlag, 2008.

[9] M. Radovanović, A. Nanopoulos, and M. Ivanović, "Nearest neighbors in high-dimensional data: The emergence and influence of hubs," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, (New York, NY, USA), pp. 865–872, ACM, 2009.

[10] R. Dutter and P. Filzmoser, *Statistik und Wahrscheinlichkeitsrechnung fuer Informatiker*. 2011 ed. Tu Wien.

[11] M. D. Smucker, J. Allan, and B. Carterette, "A comparison of statistical significance tests for information retrieval evaluation," in *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, (New York, NY, USA), pp. 623–632, ACM, 2007.

[12] D. Schnitzer, A. Flexer, M. Schedl, and G. Widmer, "Local and global scaling reduce hubs in space," *Journal of Machine Learning Research*, vol. 13, pp. 2871–2902, 2012.

[13] S. Tsumoto and S. Hirano, "Formal analysis of leave-one-out methods based on decremental sampling scheme," in *Proceedings of the 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT) - Volume 02*, WI-IAT '14, (Washington, DC, USA), pp. 371–378, IEEE Computer Society, 2014.

[14] "Crossvalidation." `http://de.mathworks.com/help/stats/crossval.html`. Accessed: 2015-11-16.

[15] "hubtoolbox." `http://www.ofai.at/research/impml/projects/highdim.html`. Accessed: 2015-09-09.

[16] "Hypersphere toolbox." `http://www.mathworks.com/matlabcentral/fileexchange/5397-hypersphere`. Accessed: 2015-10-06.

[17] "nsphere projection." `http://stackoverflow.com/questions/3278424/how-to-generate-a-net-on-a-8-dimensional-sphere`. Accessed: 2015-10-29.

[18] P. Filzmoser, *Vorlesungsskript Daten Analyse*. 2013 ed. Tu Wien.

[19] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis, "Modeling wine preferences by data mining from physicochemical properties," *Decision Support Systems*, vol. 47, no. 4, pp. 547–553, 1998.

[20] P. W. Frey and D. J. Slate, "Letter recognition using holland-style adaptive classifiers," *Machine learning*, vol. 6, no. 2, pp. 161–182, 1991.

[21] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

List of Tables

List of Figures

```
function [R,Drem, Dproj,Dpca] = hubness_analysis(D, classes, vectors)
% Performs a quick hubness analysis with all the functions provided in this
% toolbox.
%
% This file is part of the HUB TOOLBOX available at
% http://ofai.at/research/impml/projects/hubology.html
% (c) 2013, Dominik Schnitzer <dominik.schnitzer@ofai.at>
%
% Usage:
%  hubness_analysis() - Loads the example data set and performs the
%     analysis
%
%  hubness_analysis(D, classes, vectors) - Uses the distance matrix D (NxN)
%     together with an optional class labels vector (classes) and the
%     original (optional) data vectors (vectors) to perform a full hubness
%     analysis
tic
clc

haveClasses = false;
haveVectors = false;
if (nargin == 0)
    haveClasses = true;
    haveVectors = true;
elseif (nargin == 1)
    % all ok, just analyze D
elseif (nargin == 2)
    haveClasses = true;
else
    haveClasses = true;
    haveVectors = true;
end

plotdatahistogram(D);


R = zeros(4,4); %m,n
Drem = cell(4,1);
n = size(D,1);

%Dproj=D-mean(D(:));
%Dproj=Dproj/proj(Dproj(:));

%Dproj = s + r*(p-s)/(norm(p-s))
%Dproj = mean(D(:))+5.*(D-mean(D(:)))./(norm(D-mean(D(:))));

Dproj = HyperSphere([D]);

[pc,score,latent,tsquare] = princomp(D);
var = cumsum(latent)./sum(latent);
i = 3;
while var(i) < 0.8
    i = i + 1;
```

```
end
Dpca = score(:,1:i);




%C = @(D)(0.5 * (1 + erf(D/sqrt(2))));
%Dstd = C(D);            % Approximately uniform

%plotdatahistogram(Dstd);

%Dstd = (D-min(D(:))) ./ (max(D(:)-min(D(:))));

%matlabpool ('open',2);

j=1;
for i={'L1','L2','Linf','cosine'}

    [Sn5, tmp, Nk5] = hubness_ondemand(D, 5,i{1});  %datenmatrix
    R(1,j)=Sn5;


    fprintf('\nHubness Analysis\n\n');
    fprintf('ORIGINAL DATA:\n');
    fprintf('data set hubness (S^n=5)                : %.2f\n', Sn5);
    fprintf('%% of anti-hubs at k=5                   : %.2f%%\n',...
        100*sum(Nk5==0)/n);
    fprintf('%% of k=5-NN lists the largest hub occurs: %.2f%%\n',...
        100*max(Nk5)/n);



    %sortedValues = unique(Nk5(:));          %# Unique sorted values
    %maxValues = sortedValues(end-(length(D)/100-1):end)  %# Get the 1% largest values
    %maxIndex = ismember(Nk5,maxValues)     %# Get a logical index of all values
    %#   equal to the 10 largest values

    [sortedX,sortingIndices] = sort(Nk5(:),'descend');
%    maxValues = sortedX(1:(length(D)/100-1));
    maxIndex = logical(zeros(length(D),1));
    maxIndex(sortingIndices(1:((length(D)/100)-1)))=1;




    %plotpca(pc,score,maxIndex);



    [Dremoved,PS] = removerows(D,'ind',maxIndex);
```

```
    Drem{j}=maxIndex;

    [Sn5, tmp, Nk5] = hubness_ondemand(Dremoved, 5,i{1});  %datenmatrix
    R(2,j)=Sn5;



    fprintf('\nHubness Analysis after exclution\n\n');
    fprintf('data set hubness (S^n=5)                : %.2f\n', Sn5);
    fprintf('%% of anti-hubs at k=5                   : %.2f%%\n',...
        100*sum(Nk5==0)/n);
    fprintf('%% of k=5-NN lists the largest hub occurs: %.2f%%\n',...
        100*max(Nk5)/n);


    [Sn5, tmp, Nk5] = hubness_ondemand(Dproj, 5,i{1});  %datenmatrix
    R(3,j)=Sn5;

    fprintf('\nHubness Analysis after projection\n\n');
    fprintf('data set hubness (S^n=5)                : %.2f\n', Sn5);
    fprintf('%% of anti-hubs at k=5                   : %.2f%%\n',...
        100*sum(Nk5==0)/n);
    fprintf('%% of k=5-NN lists the largest hub occurs: %.2f%%\n',...
        100*max(Nk5)/n);
    fprintf('\n');


        [Sn5, tmp, Nk5] = hubness_ondemand(Dpca, 5,i{1});  %datenmatrix
    R(4,j)=Sn5;

    fprintf('\nHubness Analysis after PCA\n\n');
    fprintf('data set hubness (S^n=5)                : %.2f\n', Sn5);
    fprintf('%% of anti-hubs at k=5                   : %.2f%%\n',...
        100*sum(Nk5==0)/n);
    fprintf('%% of k=5-NN lists the largest hub occurs: %.2f%%\n',...
        100*max(Nk5)/n);
    fprintf('\n');



    j=j+1;
end




plothubness(R);
toc


%matlabpool ('close');
```

```
end



function plotdatahistogram(x)
    figure
    hist(x)
    title('letterrecognition');
    %xlabel('neighbors');
    %ylabel('Objects');

end

function plotpca(pc,score,maxIndex)
    figure
    hbi = biplot(pc(:,1:2),'Scores', score(:,1:2),'ObsLabels', num2str(maxIndex),'markersiz
    title('Principal Component Analysis');
    xlabel('PCA1');
    ylabel('PCA2');

    for ii = 1:length(hbi)
    userdata = get(hbi(ii), 'UserData');
        if ~isempty(userdata)
            if maxIndex(userdata) == 0
            set(hbi(ii), 'Color', 'g', 'Marker', 's');

            elseif maxIndex(userdata) == 1
            set(hbi(ii), 'Color', 'r', 'Marker', 's');
            end

        end
    end
end


function plothubness(R)

figure
h=bar(R);
colormap summer
ylim([-1 5])

% Add a legend
legend('L1', 'L2', 'Linf', 'cosine')

% Add title and axis labels
title('Hubness result')
ylabel('skewness of n-occurence')
xlabel = {'original'; 'exclusion'; 'sphere'; 'PCA'};
set(gca, 'XTickLabel',xlabel, 'XTick',1:numel(xlabel))
```

```
yb = cat(1, h.YData);
xb = bsxfun(@plus, h(1).XData, [h.XOffset]');

hold on;

for i = 1:size(yb,2)
   for j = 1:4
        text(xb(j, i),yb(j, i), cellstr(num2str(R(i, j))), 'rotation', 90);
   end
end


end

function barcomp(R1,R2,R3)
figure
bar(R1,'g')
ylim([-1 5])
hold on
bar(R2,0.4,'r')
ylim([-1 5])
hold on
bar(R3,0.1,'b')
ylim([-1 5])

% Add title and axis labels
title('Hubness result')
ylabel('skewness of n-occurence')
xlabel = {'original'; 'exclusion'; 'sphere'; 'PCA'};
set(gca, 'XTickLabel',xlabel, 'XTick',1:numel(xlabel))
title('Hubness result difference calculated/rangesearch')
end



function [Sn, Dk, Nk] = hubness_ondemand(D, k, dist, isSimilarityMatrix)
% Computes the hubness of a distance matrix using its k nearest neighbors.
% Hubness [1] is the skewness of the n-occurrence histogram.
%
% This file is part of the HUB TOOLBOX available at
% http://ofai.at/research/impml/projects/hubology.html
% (c) 2013, Dominik Schnitzer <dominik.schnitzer@ofai.at>
%
% Usage:
%   Sn = hubness(D) - Computes the hubness (Sk) of the n=5 occurrence histogram
%      (standard)
%
%   [Sn, Dk, Nk] hubness(D, k) - Computes the hubness of the n-occurrence
%      histogram where n (k) is given. Nk is the n-occurrence histogram, Dk
%      are the k nearest neighbors.
%
% [1] Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data
% Radovanovic, Nanopoulos, Ivanovic, Journal of Machine Learning Research 2010
```

```
    if (nargin < 2)
        k = 5;
    end
    if (nargin < 4)
        isSimilarityMatrix = 0;
    end

%     if (isSimilarityMatrix == 1)
%         d_self = -Inf;
%         sortorder = 'descend';
%     else
%         d_self = +Inf;
%         sortorder = 'ascend';
%     end


   tic
   %Dk = distance_onepoint(D,dist,k);
   %Dk = distance_memfile(D,dist,k);


   Dk = distance_range(D,dist,k);
   %Dk = distance_range_onepoint(D,dist,k);

   %disp(Dk(1:5,1:5))
    toc

    Nk = zeros(length(D), 1);
    for i = 1:length(D)
        Nk(i,1) = sum(sum(Dk == i));
    end

    Sn = local_skewness(Nk);
end

function s = local_skewness(x)
    x0 = x -  mean(x);
    s2 = mean(x0.^2);
    m3 = mean(x0.^3);
    s = m3 ./ s2.^(1.5);
end



function Dk = distance_onepoint(D,dist,k)
 if (nargin == 0)
        %fehlerfall!!
        fprintf('ERROR\n');
    elseif (nargin == 1)
        % all ok, use default
        dist='L2'
    elseif (nargin == 2)
        % all ok
 end
```

```
    d_self = +Inf;
    sortorder = 'ascend';
    Dk = zeros(k, size(D, 1));  %replaced since distmatrix is replaced with datamatrix
    for i = 1:size(D, 1)
        % extract distance matrix row %calculate
        %d = D(i, :); %original

        d = distance(D(i,:),D,dist); %on demand

            if ~(mod(i,10000))
                disp(i)
            end

        % make non-finite (NaN, Inf) appear on the end of the sorted list
        d(~isfinite(d)) = d_self;
        d(i) = d_self;

        % randomize the distance matrix row to avoid the problem case
        % if all numbers to sort are the same, which would yield high
        % hubness, even if there is none.
        rp = randperm(size(D, 1)); %dimension changed
        d2 = d(rp);
        [tmp, d2idx] = sort(d2, sortorder);
        Dk(:, i) = rp(d2idx(1:k));

     end



end

function d = distance(p,D,dist)
    switch dist
        case 'L1'
            d = pdist2(p,D,'cityblock');
        case 'L2'
            d = pdist2(p,D,'euclidean');
        case 'Linf'
            d = pdist2(p,D,'chebychev');
        case 'cosine'
            d = pdist2(p,D,'cosine');
        otherwise

    end
end



function Dk = distance_range(D,dist,k)

range=5;
%range=size(D, 1)/1000*0.5;
```

```
counter=1;
reidx = ones(size(D,1),1);
Dreidx=D;
while 1
    disp(counter)
    counter=counter+1;
    switch dist
        case 'L1'
            [idx] = rangesearch(D,Dreidx,range,'Distance','cityblock');
        case 'L2'
            [idx] = rangesearch(D,Dreidx,range,'Distance','euclidean');
        case 'Linf'
            [idx] = rangesearch(D,Dreidx,range,'Distance','chebychev');
        case 'cosine'
            [idx] = rangesearch(D,Dreidx,range,'Distance','cosine');
    end

    %recieves how many neighbors were found for each point
    cellsz = cellfun(@size,idx,'uni',false);

    %how many points have more then k neighbors calculated
    reidx = ones(size(D,1),1);
    for i = 1:size(D, 1)
        if cellsz{i}(2)>k+1
            reidx(i)=0;
        end
    end

    if sum(reidx)<(size(D,1)/100*50) %if more than 50% is filled
        break;
    elseif sum(reidx)>(size(D,1)/100*80)%if less then 80% is filled
        range=range+30;
    elseif sum(reidx)>(size(D,1)/100*60)
        range=range+5;
    end
end


%[Dreidx,PS] = removerows(D,'ind',reidx);

Dk = zeros(k+1, size(D, 1));

for i = 1:size(D, 1)
    try
        Dk(:, i) = idx{i,1}(1:k+1);
    catch exception
    end

end
Dk(1,:)=[];



function Dk = distance_range_onepoint(D,dist,k)
```

```
range=size(D, 1)/1000*0.5;
Dk = zeros(k+1, size(D, 1));

Mdlcity = KDTreeSearcher(D);
Mdlcity.Distance = 'cityblock';

Mdle = KDTreeSearcher(D);

Mdlcheby = KDTreeSearcher(D);
Mdlcheby.Distance = 'chebychev';

for i = 1:size(D, 1)
        switch dist
            case 'L1'
                [idx] = rangesearch(Mdlcity,D(i,:),range,'Distance','cityblock');
            case 'L2'
                [idx] = rangesearch(Mdle,D(i,:),range,'Distance','euclidean');
            case 'Linf'
                [idx] = rangesearch(Mdlcheby,D(i,:),range,'Distance','chebychev');
            case 'cosine'
                [idx] = rangesearch(D,D(i,:),range,'Distance','cosine');
        end

        %recieves how many neighbors were found for each point
        cellsz = cellfun(@size,idx,'uni',false);

        %if mor then k neighbors were found
        if cellsz{1}(2)>k+1
                Dk(:, i) = idx{1}(1:k+1);
        else
            i=i-1; %if not do again with adjusted range
        end

        %adjust range
        if cellsz{1}(2)<k+1
            range=range+5;
        elseif cellsz{1}(2)>(size(D,1)/100*10)
            range=range-10;
            if range<0
                range=1;
            end
        end


end
Dk(1,:)=[];



end
```

```
function Y = HyperSphere(X)
X=X';
lx = repmat(sqrt(sum(X.^2,1)), [size(X,1) 1]);
Y = X./lx;
Y=Y';
%for j = 1:size(X,2)
%norm(Y(j,:)) % Y(j,:) is the jth point on the circle)
%end

end



function crossvalidation(X,y,Drem,Dpca,Dproj)

whos
%X=D(:,1:11);
%y=int2str(y);

fprintf('original:\n');
computeCV(X,y)  % 0.7062
%computecf(X,y)

for i = 1:4
  [Xrem,PS] = removerows(X,'ind',Drem{i});
  [yrem,PS] = removerows(y,'ind',Drem{i});
%  yrem=int2str(yrem);
  fprintf('removed rows:\n');
  computeCV(Xrem,yrem)
end

X=Dpca;
fprintf('pca:\n');
computeCV(X,y)  % 0.8056

X=Dproj;
fprintf('projection:\n');
computeCV(X,y) %0.9982







end

function cvMCR = computeCV(X,y)
cp = cvpartition(y,'k',10); % Stratified cross-validation

classf = @(XTRAIN, ytrain,XTEST)(classify(XTEST,XTRAIN,...
ytrain));


cvMCR = crossval('mcr',X,y,'predfun',classf,'partition',cp);
```

```
end

function cfMat = computecf(X,y)

order = unique(y) % Order of the group labels

cp = cvpartition(y,'k',10); % Stratified cross-validation

f = @(xtr,ytr,xte,yte)confusionmat(yte,...
classify(xte,xtr,ytr),'order',order);

cfMat = crossval(f,X,y,'partition',cp);
cfMat = reshape(sum(cfMat),length(order),length(order));
end
```