PAMI homework 2: Curse of Dimensionality

Egor Makhov 10493074

```
library(ggplot2)
library(grid)
```

```
rm(list=ls())

# set the seed as the last three digits of your student ID
seed = 074

if (seed == 111) {
  stop()
}
set.seed(seed)
```

```
vol_of_sphere <- function(dim, N) {
  m <- matrix(runif(dim*N,-1,1), ncol=dim)
  sum(sqrt(rowSums(m^2)) <= 1) / N
}

rand_dist_origin <- function(N,dim) {
  m <- matrix(rnorm(N*dim),ncol=dim)
  v <- sqrt(rowSums(m^2))
  return(c(avg = mean(v), min = min(v), max = max(v), rel = ((max(v)-min(v))/min(v))))
}

perform_lda_prediction <- function(fit, data, labels) {
  pred = predict(fit, as.data.frame(data))
  class = pred$class
  table(class,labels)
  mean(class == labels)
}
```

Welcome to the second PAMI demo/homework (year 2016): curse of dimensionality

The "curse of dimensionality" is a phenomenon (or, better, many phenomena) occurring when the dimensionality of your problem, e.g. the number of features that characterizes your observation, increases. When this happens, data tends to become sparse, making it difficult for any method which requires statistical significance to work properly.

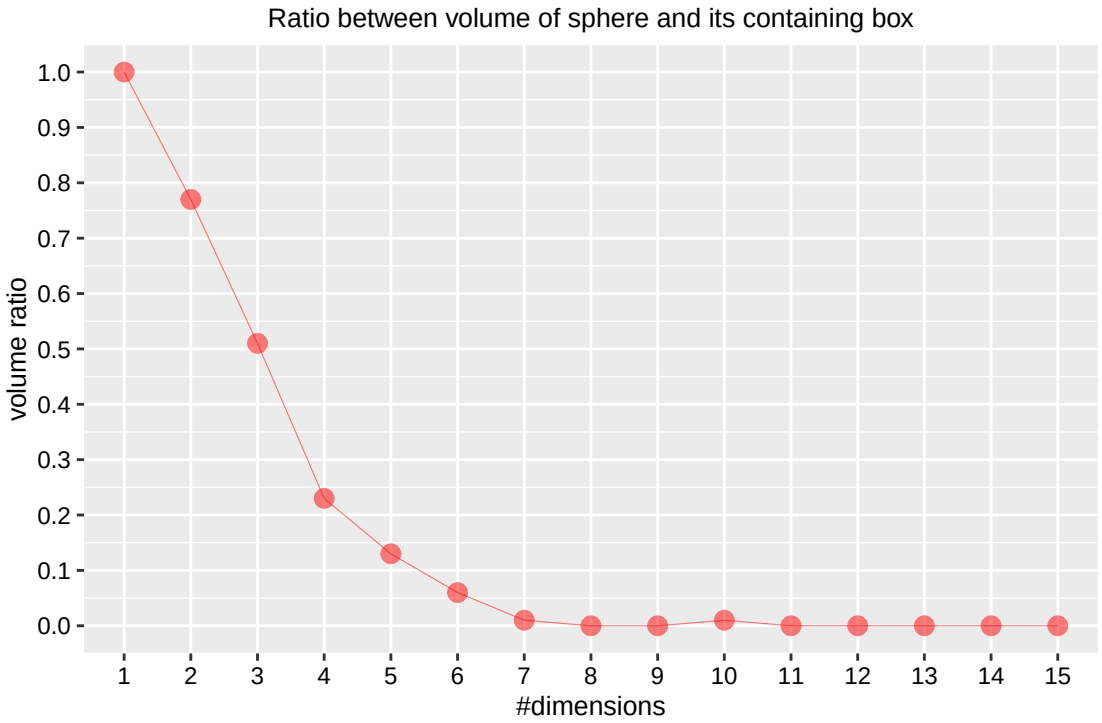
Let us try few examples from http://www.joyofdata.de/blog/curse-dimensionality/ and comment them.

```
N = 100 # this is the number of points filling the hyperbox, decrease if too slow

df <- data.frame(
  dim <- 1:15,
  v <- sapply(1:15, function(dim) vol_of_sphere(dim,N))
)

ggplot(df,aes(dim,v)) +
  geom_point(colour=I("red"), size=I(3), alpha=I(0.5)) +
```

```
  geom_line(colour=I("red"), size=I(.2), alpha=I(0.5)) +
  scale_x_discrete() +
  labs(title="Ratio between volume of sphere and its containing box",
       x="#dimensions",
       y="volume ratio") +
  scale_y_continuous(breaks=0:10/10) +
  theme(axis.text.x = element_text(colour="black"),
        axis.text.y = element_text(colour="black"),
        axis.title.x = element_text(vjust=-0.5, size=10),
        axis.title.y = element_text(vjust=-0.2, size=10),
        plot.margin = unit(c(1,1,1,1), "cm"),
        plot.title = element_text(vjust=2, size=10))
```



This first example shows how the ratio between the volume of a (hyper)sphere of radius 1 and its containing (hyper)box when the number of dimensions increases. Answer the following questions:

Q1) How is this ratio calculated? Look at the code and tell how the volumes of the sphere and the hypercube are found.

We are generating N points inside our hyper box using Uniform distribution. After that we are counting how many points are inside the hypersphere (center of the sphere is the center of our n-dimensional space so if euclidian distance of the point from begining of coordinates is =< 1 => the point is in the hyper sphere).

Just to keep in mind: The volume of a hypercube of dimension dim is edgeˆdim. The volume of the inscribing hypersphere of dimension d and with radius 0.5 can be calculated as $V(d) = \frac{\pi^{d/2}}{\Gamma(\frac{d}{2}+1)} r^d$

Q2) What is the actual meaning of this plot? What does it mean for a point in the box to fall or not within the sphere too?

This plot shows that the volume of the hypersphere tends to zero as the dimensionality tends to infinity, whereas the volume of the surrounding hypercube remains almost constant.

The effect is that with the increase of dimension, points are rarely any longer close to points we would (interpreting the data) consider similar (meaning "living in the same neighbourhood").

One of the main concepts whose interpretation is affected by the curse of dimensionality is the one of *similarity*, that here is dual to distance.

Q3) Check out the code that generates the following results. What does it do? Modify the "dim" variable and comment the results of the summary() command

```
dim = 3
m <- matrix(runif(dim*N,-1,1), ncol=dim)
d = sqrt(rowSums(m^2))
summary(d)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.2318  0.8220  0.9984  0.9790  1.1760  1.5120
```

```
dim = 7
m <- matrix(runif(dim*N,-1,1), ncol=dim)
d = sqrt(rowSums(m^2))
summary(d)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.6652  1.3180  1.5000  1.4700  1.6420  2.0050
```

```
dim = 9
m <- matrix(runif(dim*N,-1,1), ncol=dim)
d = sqrt(rowSums(m^2))
summary(d)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8266  1.5510  1.7080  1.7210  1.9290  2.2900
```

```
dim = 15
m <- matrix(runif(dim*N,-1,1), ncol=dim)
d = sqrt(rowSums(m^2))
summary(d)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.407   2.035   2.187   2.190   2.378   2.662
```

This code calculating Euclidian distance from the begining of coordinates to the points (generated with Uniform Distribution).

When dimensionality increases the min and max increases too, but (max-mean) stays almost the same.

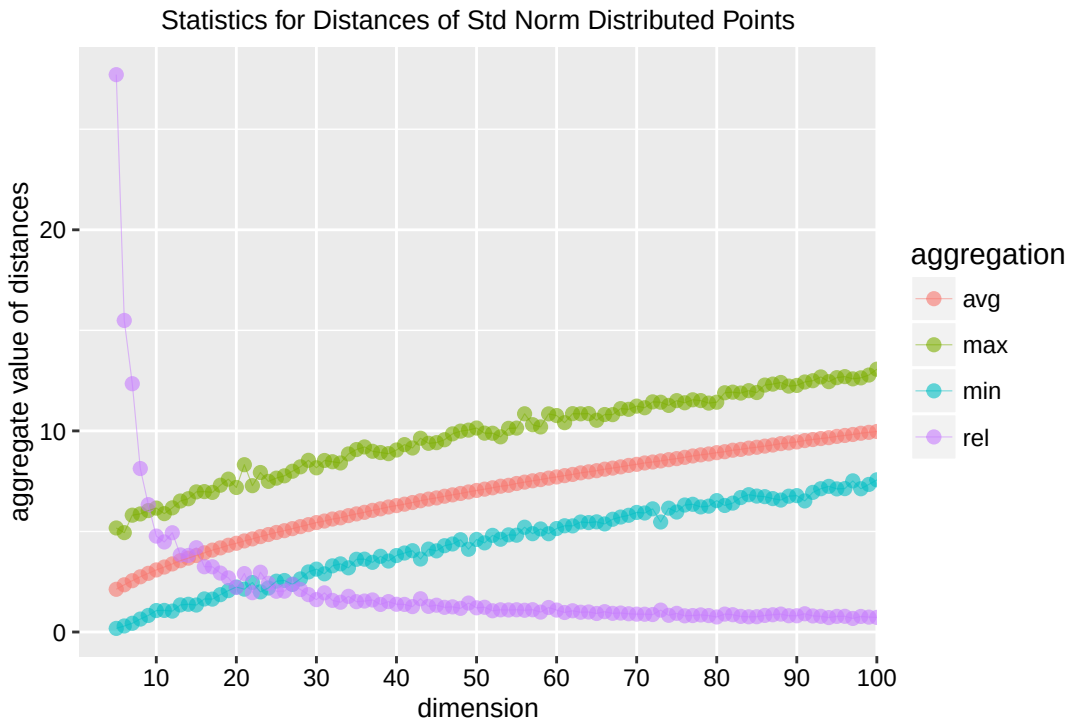
Let us perform the same operation more consistently, generating random datasets with increasing dimensionality and checking how min, max, and average distance change.

```
max_dim <- 100
N = 10000

v <- mapply(rand_dist_origin, rep(N,max_dim-4), 5:max_dim)

df <- data.frame(
  dim = 5:max_dim,
  aggregation = as.factor(c(rep("min", length(v["min",])), rep("max", length(v["max",])), rep("avg", le
  v = c(v["min",], v["max",], v["avg",], v["rel",])
)

ggplot(df,aes(dim,v)) +
  geom_point(aes(colour=aggregation), size=I(2), alpha=I(0.6)) +
  geom_line(aes(colour=aggregation), size=I(.2), alpha=I(0.5)) +
  scale_x_discrete(breaks=1:10*10) +
  labs(title="Statistics for Distances of Std Norm Distributed Points",
       x="dimension",
       y="aggregate value of distances") +
  scale_y_continuous(breaks=0:21*10) +
  theme(axis.text.x = element_text(colour="black"),
        axis.text.y = element_text(colour="black"),
        axis.title.x = element_text(vjust=-0.5, size=10),
        axis.title.y = element_text(vjust=-0.2, size=10),
        plot.margin = unit(c(1,1,1,1), "cm"),
        plot.title = element_text(vjust=2, size=10))
```



Q4) What happens when dimensionality increases? Note that the difference between max and min roughly remains constant... But what happens to the *relative difference* (i.e. (max-min)/min)? Try to plot it together with the other data and explain the meaning of this result.

I added relative difference of data to the previous plot

As the plot indicates the relative difference between the maximum observed distance and the minimum observed distance vanishes with increase of dimension – this means variations in distance convey less and less information on similarity and this handicaps clustering using methods like k-NN (assuming it applies the euclidian distance).

$\lim_{x\to\infty} \frac{dist_{max} - dist_{min}}{dist_{min}} \to 0$

Q5) Finally, consider a classification problem (e.g. a set of 100 "email" observations you need to classify as either "spam" or "not spam") and explain how dimensionality might affect its results. Given the same amount of observations, is it better to have more or less features? What happens when you have many of them? How can you address potential problems due to the curse of dimensionality?

Consider an example in which we have a set of images, each of which depicts either a lemon or an apple. We would like to create a classifier that is able to distinguish apples from lemons automatically. A possible descriptor that discriminates these two classes could then consist of three number; the average red color, the average green color and the average blue color of the image under consideration. However, these three color-describing numbers, called features, will obviously not suffice to obtain a perfect classification. Therefore, we could decide to add some features that describe the texture of the image, for instance by calculating the average edge or gradient intensity in both the X and Y direction. We have 5 features.

Now let's assume that we were only able to obtain 10 pictures to classify. The end-goal in classification is then to train a classifier based on these 10 training instances, that is able to correctly classify the infinite number of pictures.

In the 1D case, 10 training instances covered the complete 1D feature space, the width of which was 5 unit intervals. Therefore, in the 1D case, the sample density was 10/5=2 samples/interval. In the 2D case however, we still had 10 training instances at our disposal, which now cover a 2D feature space with an area of 5×5=25 unit squares. Therefore, in the 2D case, the sample density was 10/25 = 0.4 samples/interval. Finally, in the 3D case, the 10 samples had to cover a feature space volume of 5x5x5=125 unit cubes. Therefore, in the 3D case, the sample density was 10/125 = 0.08 samples/interval.

If we would keep adding features, the dimensionality of the feature space grows, and becomes sparser and sparser. Due to this sparsity, it becomes much more easy to find a separable hyperplane because the likelihood that a training sample lies on the wrong side of the best hyperplane becomes infinitely small when the number of features becomes infinitely large. However, if we project the highly dimensional classification result back to a lower dimensional space, a serious problem associated with this approach becomes evident: Using too many features results in overfitting. The classifier starts learning exceptions that are specific to the training data and do not generalize well when new data is encountered.

In other words, if the amount of available training data is fixed, then overfitting occurs if we keep adding dimensions. On the other hand, if we keep adding dimensions, the amount of training data needs to grow exponentially fast to maintain the same coverage and to avoid overfitting.
