Dimensionality reduction

Diem Huong Nguyen

17 listopadu 2019

Introduction

The goal of this tutorial is to get familiar with some basic methods for dimensionality reduction, complete you own implementation of the **Isomap algorithm** (in cooperation with *Multidimensional scaling*), experiment with its parameters and compare with other techniques of dimensionality reduction (**PCA**, **t-SNE**).

Background

The data you will be working with are vector representations of words in a latent (unknown) high-dimensional space. This representation of words, also know as word embedding, differs from standard bag-of-words (BoW, TFIDF, etc.) representations in that the meaning of the words is distributed across all the dimensions. Generally speaking, the word embedding algorithms seek to learn a mapping projecting the original BoW representation (simple word index into a given vocabulary) into a lower-dimensional (but still too high for our cause) continuous vector-space, based on their distributional properties observed in some raw text corpus. This distributional semantics approach to word representations is based on the basic idea that linguistic items with similar distributions typically have similar meanings, i.e. words that often appear in a similar context (words that surround them) tend to have similar (vector) representations.

Speciffically, the data you are presented with are vector representations coming from the most popular algorithm for word embedding known as word2vec[1] by Tomas Mikolov (VUT-Brno alumni). *Word2vec* is a (shallow) neural model learning the projection of BoW word representations into a latent space by the means of gradient descend. Your task is to further reduce the dimensionality of the word representations to get a visual insight into what has been learned.

Data

You are given 300-dimensional word2vec vector embeddings in the file *data.csv* with corresponding word labels in *labels.txt* for each line. Each of these words comes from one of 10 selected classes of synonyms, which can be recognized (and depicted) w.r.t. labels denoted in the file *colors.csv*

Tasks

1. **Load the dataset of 165 words**, each represented as a 300-dimensional vector. Each word is assigned to one of 10 clusters.

```
#Load the dataset of 165 words
mydata <- read.csv('data.csv', header = FALSE)
mylabels <- read.csv('labels.txt', header = FALSE)
mycolors <- read.csv('colors.csv', header = FALSE)
```

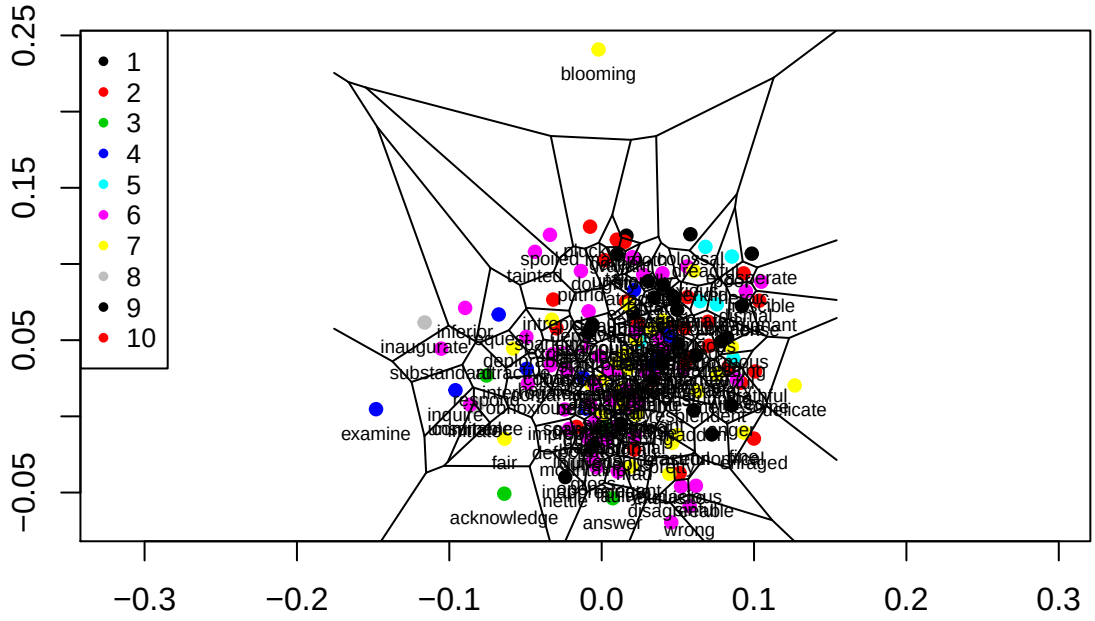
The data is in the matrix `mydata`, cluster assignment in `mycolors` and the actual words (useful for visualization) in `mylabels`. You can plot the data by using only the first 2 dimensions.

```
PlotPoints(mydata[,c(1,2)], mylabels, mycolors)
```

```
## deldir 0.1-23
```

[1] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111-3119, 2013.

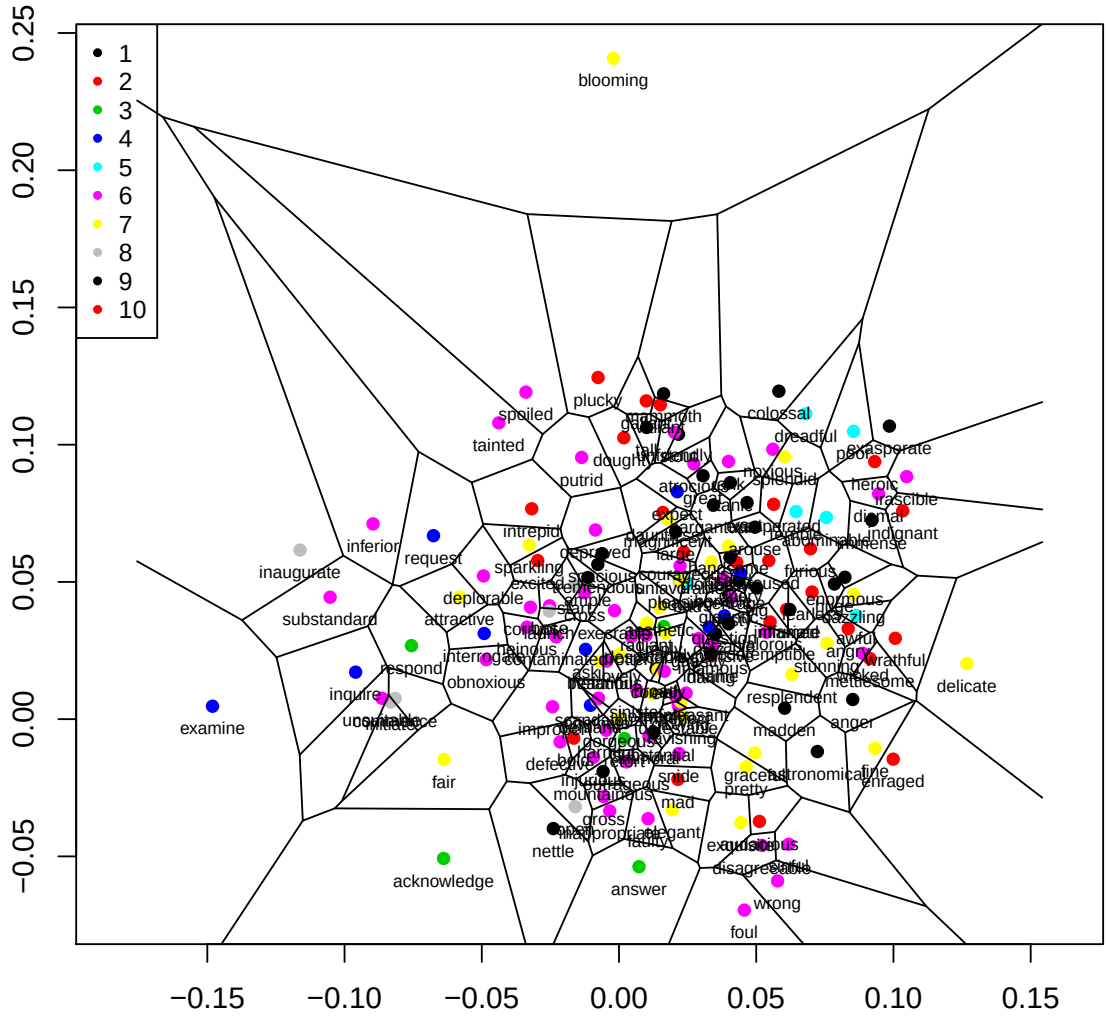
```
##
##      PLEASE NOTE:  The components "delsgs" and "summary" of the
##  object returned by deldir() are now DATA FRAMES rather than
##  matrices (as they were prior to release 0.0-18).
##  See help("deldir").
##
##      PLEASE NOTE: The process that deldir() uses for determining
##  duplicated points has changed from that used in version
##  0.0-9 of this package (and previously). See help("deldir").
```



2. **Implement ISO-MAP dimensionality reduction procedure**.

- Use k-NN method to construct the neighborhood graph (sparse matrix).
  - For simplicity, you can use `get.knn` method available in `FNN` package.
- Compute shortest-paths (geodesic) matrix using your favourite algorithm.
  - Tip: Floyd-Warshall algorithm can be implemented easily here.
- Project the geodesic distance matrix into 2D space with (Classical) Multidimensional Scaling (`cmdscale` functions in R).
- Challenge: you may simply use PCA to do the same, but be careful to account for a proper normalization (centering) of the geodesic (kernel) matrix (see Kernel PCA for details).

An expected result (for $k = 5$) should look similar (not necessarily exactly the same) to following




```
#ADD YOUR CODE HERE!
#REMOVE eval=FALSE arguments
library(FNN)
#1.Determine the neighbors of each point (k-NN)
rows <- nrow(mydata)

myfriends <- as.matrix(c(1:rows))
knn <- get.knn(mydata, k = rows - 1)
index <- cbind(myfriends, knn[["nn.index"]])
distances <- as.matrix(knn[["nn.dist"]])

#2.Construct a neighborhood graph.
#Each point is connected to other if it is a K nearest neighbor.
#Edge length equal to Euclidean distance.
```

```
output <- matrix(Inf, nrow=nrow(index), ncol=ncol(index))
k = 5
for (i in 1:rows) {
  for (j in 1:k) {
    output[i, i] = 0
    neigh_index = index[i,j + 1]
    output[i, neigh_index] <- distances[i, j]
    output[neigh_index, i] <- distances[i, j]
  }
}

#3.Compute shortest path between two nodes. (Floyd-Warshall algorithm)

# Vlastni floyd
#for (k in 1:rows) {
#    for (i in 1:rows) {
#      for (j in 1:rows) {
#       new = output[i, k] + output[k, j]
#        if (output[i, j] >  new) {
#          output[i, j] = new
#        }
#      }
#    }
#}

# Funkce floyd
D <- floyd(output)
#4.(classical) multidimensional scaling
fit <- cmdscale(D,eig=TRUE, k=2)
PlotPoints(fit$points, mylabels, mycolors)
```

3. **Visually compare PCA, ISOMAP and t-SNE** by plotting the word2vec data, embedded into 2D using the `Plotpoints` function. Try finding the optimal k value for ISOMAP's nearest neighbour.

```
# Visual comparison

Metoda multidimenzionalního škálování byla schopna dobře shluknout růžová (negativní adjektiva) a žlutá

PCA byla schopna shluknout žlutá slova, která jsou ovšem více roztroušená, než u předchozí metody. Dále

T-SNE bohužel nevytvořilo nic, co by se dalo přímo nazávat shlukem. Je vidět, že se tečky stejné barvy z

růžová - záporný adjektiva
zelená - interakce
černá - adjektiva popisující velikost
žlutá - pozitivní adjektiva
šedivá - slovesa naznačující začátek
tyrkysová - taky negativní adjektiva
modrá - slovesa naznačující nějakou mozkovou aktivitu
červená - citově zabarvená negativní adjektiva



#Principal component analysis
```

```
fitPCA <- prcomp(mydata, center = TRUE, scale. = TRUE)
PlotPoints(fitPCA$x[,c(1,2)], mylabels, mycolors)
```

```
#t-SNE (T-Distributed Stochastic Neighbor Embedding)
#install.packages('tsne')
library(tsne)
fittsne <- tsne(mydata, k = 2)
PlotPoints(fittsne, mylabels, mycolors)
```

4. **Observe the effect of dimensionality reduction on a classiffication algorithm**. The supporting code in a function `Classify` performs training and testing of classification trees and gives the classification accuracy (percentage of correctly classified samples) as its result. Compare the accuracy of prediction on plain data, PCA, ISOMAP and t-SNE.

```
#classify ISOMAP
accPlainData <- Classify(mydata, mycolors$V1, 50)

accISOMAP <- Classify(as.data.frame(fit$points), mycolors$V1, 50)

#classify PCA
accPCA <- Classify(as.data.frame(fitPCA$x), mycolors$V1, 50)

#classify t-SNE
accTSNE <- Classify(as.data.frame(fittsne), mycolors$V1, 50)

#PLOT results
print(paste("Plain data:", mean(accPlainData)))
print(paste("ISOMAP ACC:", mean(accISOMAP)))
print(paste("PCA ACC:", mean(accPCA)))
print(paste("t-SNE ACC:", mean(accTSNE)))
```