Dimensionality Reduction

WS 2015/2016

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, experiment with its parameters and compare with other techniques of dimensionality reduction.

1 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.

Specifically, 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.

2 Data

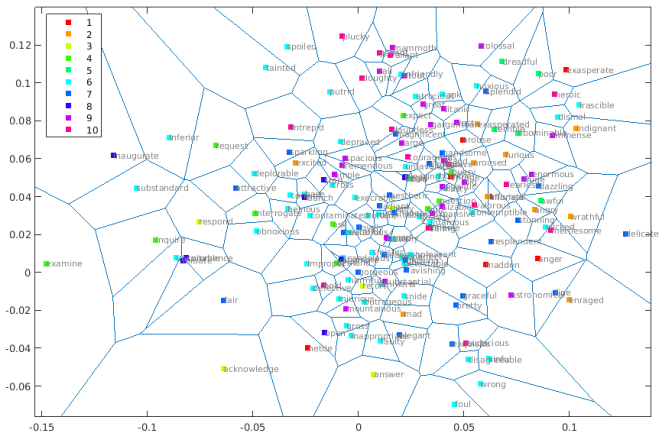
You are given 300-dimensional word2vec vector embeddings in the file *data.csv* with corresponding word labels in *labels.txt* for each line. Each of these words comes from one of 10 selected classes of synonyms, which can be recognized (and depicted) w.r.t. labels denoted in the file *colors.csv*.

3 Tasks

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

   ```
   X = load('data.csv');
   labels = textread('labels.txt', '%s', 'delimiter', '\n');
   colors = load('colors.csv');
   ```

   The data is in the matrix `X`, cluster assignment in `colors` and the actual words (useful for visualization) in `labels`. You can plot the data by using only the first 2 dimensions (Matlab: `plotpoints(X, labels, colors);`)

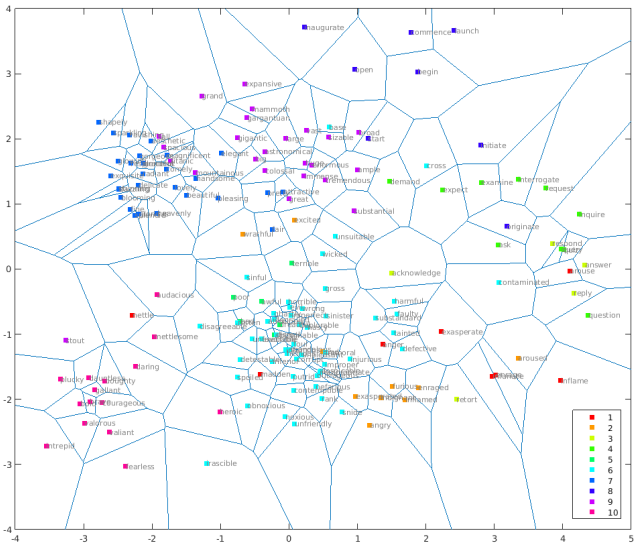


   The supporting code is provided in a file `execute.m`.

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

   - Use k-NN to construct the neighborhood graph.
   - Compute shortest-path using your favourite algorithm. Tip: In Matlab, the Floyd-Warshall algorithm can be implemented very quickly.
   - For low-dimensionality reduction, use PCA.

   The expected result (for $k = 5$) should look as follows



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.

   PCA and t-SNE is already provided to you as a part of the *Matlab Toolbox for Dimensionality Reduction*. See `execute.m` for details. This time, please do not use the ISOMAP method from the toolbox. We will know! :-)

4. **Observe the effect of dimensionality reduction on a classification algorithm.** The supporting code in `classify.m` 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.

References

[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.