TDA 231 Machine Learning: Homework 6

Goal: k-means clustering
Grader: Hans Samuelsson

Due Date: March 7, 2016

General guidelines:

1. All solutions to theoretical problems, and discussion regarding practical problems, should be submitted in a single file named *report.pdf*
2. All matlab files have to be submitted as a single zip file named *code.zip*.
3. The report should clearly indicate your name, personal number and email address
4. All datasets can be downloaded from the course website.
5. All plots, tables and additional information should be included in *report.pdf*

1 Practical problems

Problem 1.1 [k-Means Implementation, 8 points]

(a) Implement the basic (linear) k-means algorithm as described in the lecture, using the euclidean distance. Use (uniformly) random points from the data as initialization for the centroids. Terminate the iterative procedure when the the cluster assignments do not change.

(b) Run your implementation on the matrix X in *hw6_p1a.mat* with $k = 2$. Each row of the matrix is an observation, and each column is a feature. Store the cluster assignment both after 2 iterations, and at convergence. Produce a scatter plot of the data with colors indicating the cluster assignments at convergence and highlight points that have changed assignment after the second iteration. Include the plot in your report.

(c) Implement the kernel k-means algorithm as described in the lecture, using the Gaussian RBF-kernel.

(d) Run the linear k-means *and* your kernel k-means on *hw6_p1b.mat* with $k = 2$. For the Gaussian RBF-kernel, use $\sigma = 0.2$. Produce scatter plots of the data, with color indicating the cluster assignment at convergence, one plot for each of the algorithms. Include both plots in the report.

Problem 1.2 [k-Means Analysis, 12 points]

In this assignment you will cluster word embeddings, pre-trained using the word2vec model (see https://code.google.com/archive/p/word2vec/). Each word w in a vocabulary is assigned a d-dimensional

vector u_w which captures the semantic meaning of the word. That is, words v, w that are close in the vector space (small distance between u_w and u_v) have similar meaning. The word2vec model assumes that if two words appear with similar context (neighboring words), they can be used in the same way. You can find many fun ways of using word2vec online. As you will see, clustering word embeddings like this results in a kind of topic model.

The dataset (*medium_100_10k.mat*) that you will be using comprises the 100-dimensional embeddings of 10 000 words represented as a 10000×100 matrix called *wordembeddings*. The vocabulary is called *vocab* with $vocab\{i\}$ the word represented by row i in *wordembeddings*. For example, $vocab\{5000\} = instruction$.

Download the visualization tool tSNE: https://lvdmaaten.github.io/tsne/code/tSNE_matlab.zip and put it in the same folder as your code. You can read the word embeddings in the file *medium_100_10k.mat* using MATLAB's *load*.

(a) Run MATLAB's *kmeans* with $k = 10$, and default parameters, on the word embedding data and report the 10 words that are closest to the centroids in each cluster.

(b) Evaluate the pairwise stability of the clustering. Run kmeans in the same way as before, but with the parameter "Replicates" set to 1. Find the cluster that contains the word "cavalry", count how many word *pairs* are in that cluster, and denote that N_0. Now run kmeans again, with the same parameters. Compute the number N_1 of pairs of words that were both originally in the "cavalry"-cluster, but that are also in the same cluster in the second run (not necessarily together with "cavalry"). Now, let the fraction $f = N_1/N_0$ be a measure of cluster stability. This is an example of how to compute f. Let C be the original clustering and C' the second.

$$
\begin{aligned}
C_1 &= \{\text{cavalry, war, cannon, horse}\}, C_2 = \{\text{riding}\} \\
C_1' &= \{\text{war, cannon}\}, C_2' = \{\text{cavalry, horse, riding}\} \\
N_0 &= |\{\text{cavalry-war, cavalry-cannon, cavalry-horse, war-cannon, war-horse, cannon-horse}\}| = 6. \\
N_1 &= |\{\text{war-cannon, cavalry-horse}\}| = 2. \\
f &= 1/3
\end{aligned}
$$

Repeat the above experiment 10 times (20 kmeans runs in total), and report the average fraction of word pairs that remain in the same cluster. What can you conclude about the clustering? *Note that the clusters may not have the same index for different runs.*

(c) Visualize the word embeddings of 1000 randomly sampled words as a point cloud. First use the downloaded matlab function *tsne* to project the embeddings into 2D, and then produce a scatter plot. Assign a different color/marker to each of the computed clusters. Include the plot in your report.