`ADENINE`: A Data ExploratioN pipelINE

development plan

Samuele Fiorini

May 21, 2015

1 Introduction and Motivation

A question that arises at the beginning of almost every new data analysis is the following: *are my data relevant for the problem I'm dealing with*?

The final goal of this project (named `adenine`) is to help its user to have a glimpse of the answer of this tedious question.

In order to reach this goal, `adenine` will take advantage of machine learning and data mining techniques. The final pipeline will essentially consist of three steps:

1. **Preprocessing**: have you ever wondered what would have changed if only your data have been preprocessed in a different way? Or if data preprocessing is a good idea at all? `adenine` will offer several preprocessing procedures, such as: data centering, Min-Max scaling, standardization or normalization and allows you to compare the results of the analysis conducted with different starting point.
2. **Dimensionality Reduction** (DR): in the context of data exploration, this phase becomes particularly helpful for high dimensional data (e.g. -omics scenario). This step, generically named DR, may actually include some manifold learning (such as Isomap, Multidimensional Scaling, etc), supervised (Linear Discriminant Analysis) and unsupervised (Principal Component Analysis, kernel PCA) techniques.
3. **Clustering**: this section aims at grouping data into clusters without taking into account the class labels. Several techniques such as K-Means, Spectral or Hierarchical clustering will work on both original and dimensionality reduced data.

The final output of `adenine` will be an as compact as possible visual and textual representation of the results obtained from the pipelines made with each possible combination of the algorithms implemented at each step. As an example, referring to a pipeline built as:

$$\textit{Data normalization} \rightarrow \textit{PCA} \rightarrow \textit{K-Means}$$

the output would be something like:

- an output file containing the norm of the original variables (which has been used to coerce all the features in $[0, 1]$),
- a 2-D or 3-D scatter plot of the data projected along the principal components and the percentage of explained variance associated with each one of them,
- a pictorial representation of the data clustering results obtained with the optimum number of cluster (learned from the data).

1.1 Material for PhD progress

The study behind the implementation of `adenine` will be useful in terms of four PhD courses of my first-year work plan:

1. *A Machine Learning Crash Course* [DIBRIS] (Odone, Rosasco): `adenine` will cover a fair number of (mainly unsupervised) machine learning techniques. Hence, this course has been fundamental to acquire the statistical learning background needed to become aware of the underlying mechanisms of the algorithms.
2. *Programming Concepts in Python* [DIBRIS] (Tacchella): I plan to implement `adenine` in `Python`. Hence, most of the implementation choices will be made on the basis of the material covered in the course.
3. *Programming Complex Heterogeneous Parallel Systems* [IMATI] (Clematis, D'Agostino, Danovaro, Galizia) and the *24th Summer School on Parallel Computing* [CINECA] (Erbacci): `adenine` will present several *embarrassingly parallel workload* as well as several *isolate GPU accelerable* computations. The former PhD course and the latter school will allow me to develop the parallel computing attitude I need to implement `adenine` in an as optimized as possible way.

2 Implemented Algorithms

The implementation of nearly all the algorithms of `adenine` will refer to the `scikit-learn` python library. See the following `link` for a comprehensive list,

2.1 Preprocessing

At this step the data will be fed to the following preprocessing procedures:

0. no preprocessing: the analysis will be conducted on raw data;
1. naïve recentering: remove the mean;
2. standardization: remove the mean and scale each feature by their standard deviations, this will make the data normally distributed;
3. normalization: scale all the samples to have unit norm

In its first version `adenine` will allow the user to impute the missing values by means of the median, the mean or the most frequent value (future works are in Section 3). See the `sklearn` docs on data preprocessing for further details.

2.2 Dimensionality reduction

The following is a work-in-progress list of the techniques I plan to make available in `adenine`. The list includes algorithms that come from very different standpoint, but that have a common outcome: the estimation of a low-dimensional embedding (manifold) in which the data can be projected for visualization or further purposes.

(a) Principal Component Analysis (PCA), in its Incremental or Randomized variants in case of big data;

(b) Kernel PCA, which may come along different kernels (Gaussian, polynomial, and so on);

(c) Isomap;

(d) Locally Linear Embedding (LLE), in its modified (MLLE) or Hessian (HLLE) regularized version;

(e) Spectral Embedding (SE);

(f) Local Tangent Space Alignment (LTSA);

(g) Multidimensional Scaling (MDS), in its metric and non-metric version;

(h) t-distributed Stochastic Neighbor Embedding (t-SNE).

2.3 Clustering

On the same line, this section presents a list of the clustering techniques I plan to include in `adenine`.

(α) K-Means, in its Mini-Batch variant for big data;

(β) Affinity Propagation;

(γ) Mean Shift;

(δ) Spectral Clustering;

(ϵ) Hierarchical Agglomerative Clustering, exploring different linkage type, i.e., Ward, complete, average as well as different metrics, e.g. Euclidean, Manhattan, Minkowski, etc.;

(ζ) DBSCAN;

(η) Birch.

Several indexes to analyze the clustering performances will be included, some of them may require ground truth labels (such as Adjusted Rand Index (ARI), the Adjusted Mutual Information (AMI), the homogeneity, completeness or V measure scores), while others may evaluate the cluster compactness or the separation between clusters (such as the silhouette score).

3 Future Works

Indeed `adenine` is not meant to be an all-inclusive tool. This section, that will always be a work-in-progress, aims at mentioning all the features that are not going to be implemented in the first version of `adenine`, but that may be implemented later on.

- How can we handle missing values? `adenine` may have some statistically robust imputation tools (such as low-rank matrix completion, or collaborative filtering) in future versions;
- Kernel K-Means;
- Dictionary Learning;
- Factor Analysis;
- Non-negative Matrix Factorization;
- Outliers Detection.