ADENINE — A Data ExploratioN pIpeliNE

Samuele Fiorini SAMUELE.FIORINI@DIBRIS.UNIGE.IT
Federico Tomasi FEDERICO.TOMASI@DIBRIS.UNIGE.IT
Annalisa Barla ANNALISA.BARLA@UNIGE.IT

Department of Informatics, Bioengineering,
Robotics and System Engineering (DIBRIS)
University of Genoa
Genoa, I-16146, Italy

Editor: Editor name

Abstract

In this paper we introduce `Adenine`, a machine learning `Python` framework for data exploration. The main goal of `Adenine`, is twofold: helping researchers and data scientists achieving a first and quick overview on the main structures underlying their data and choosing the most suitable unsupervised learning pipeline for the problem at hand. This software tool encompasses state of the art techniques for: missing values imputing, preprocessing, dimensionality reduction and clustering tasks. `Adenine` exploits both process and thread level parallelism and it is capable of generating nice and clean publication-ready figures along with quantitative descriptions of the pipelines performance. `Adenine` is released under FreeBSD license and it can be downloaded from http://slipguru.github.io/adenine/.

Keywords: Data exploration, unsupervised learning, RNA-Seq gene expression

1. Introduction

Data exploration is a very insightful starting point for many data analysis projects. Researchers and data scientists are often asked to extract meaningful information from collections of complex and possibly high-dimensional data coming from heterogenous contexts. For instance, in biomedical scenarios, physicians are likely to be interested in answering some biological questions starting from observations collected from a pool of subjects enrolled in a study. Possible investigations can be: *is there any relevant stratification among subjects?* or *is it possible to discriminate between cases and controls from my observations?*. Starting from a given dataset, the information needed to answer such questions may be immediate, non-trivial to extract or even completely absent. In these situations, a preliminary data exploration step is not only good practice, but also a fundamental starting point for further and deeper investigations. To accomplish this task, several machine learning and data mining techniques were developed over the years. Among those we focus on the four most popular classes of methods: (i) missing values imputing, (ii) data preprocessing, (iii) dimensionality reduction and (iv) unsupervised clustering.

In the last few years, a fair number of data exploration softwares and libraries were released. At a very coarse grain we can group them into two sets: interactive GUI-based and non-interactive command-line applications. Of the first group, we recall *Divvy* (Lewis et al., 2013), a software tool that performs dimensionality reduction and clustering on input datasets. *Divvy* is a light framework, although its interface is designed to be Mac OS X specific, it heavily lacks in documentation, its collection of `C/C++` algorithm implementations does not cover common strategies such as kernel principal component analysis (KPCA) (Schölkopf et al., 1997) or hierarchical clustering (Friedman et al., 2001) and it does not offer strategies to perform automatic discoveries of the number of clusters. The most notable project that spans between the two families is *Orange* (Demšar et al., 2013), a data mining software suite that offers both visual programming front-end and `Python` APIs. In the context of data exploration, *Orange* can be successfully employed. However, it does not support automatic pipeline generation, hence it requires the user to manually create each pipeline. On the other hand, as of today *Orange* lacks in several nonlinear methods such as spectral embedding (Ng et al., 2002), locally linear embedding (Roweis and Saul, 2000) and spectral clustering (Shi and Malik, 2000).

We introduce `Adenine`, a command-line `Python` tool for data exploration that, starting from a set of predefined unsupervised algorithms, creates textual an graphical reports of an arbitrary number of pipelines. In this context data imputing, preprocessing, dimensionality reduction and clustering strategies are considered as building blocks for data analysis pipelines. The user is only required to specify input data and to select blocks, then `Adenine` takes care of automatically generate and run the pipelines made by all possible combinations of the selected algorithms. Every algorithm implementation of `Adenine` is inherited, or extended, from `scikit-learn` (Pedregosa et al., 2011) which is, to the best of our knowledge, the most complete machine learning open source library freely available online.

2. Tool overview

`Adenine` is developed around the concept of data analysis *pipeline*, with that we mean a sequence of the following four fundamental steps: (i) missing values imputing, (ii) data preprocessing, (iii) dimensionality reduction and (iv) unsupervised clustering (see Figure 1). For each task, a fair number of off-the-shelf algorithms are available (see Table 2), however none of such steps are mandatory and they can be skipped if not required.

Step 0: Missing values imputing. In real-world datasets, groups of entries are often missing. In order to cope with this issue, `Adenine` offers an improved version of the `Imputer` class offered by `scikit-learn`. Our extension adds a k-nearest neighbor (KNN) imputing method to the pre-existent features-wise *mean*, *median* and *most frequent* value strategies. We chose to add the KNN imputing method to the *naïve* strategies offered by `scikit-learn` because of the robustness demonstrated in the microarray reconstruction experiments described in (Troyanskaya et al., 2001).

Step 1: Data preprocessing. Collecting data from heterogenous sources may imply dealing with features lying in very different numerical ranges. This can sometimes have a negative influence on the behaviour of dimensionality reduction and clustering techniques. To tackle this issue `Adenine` offers four different strategies:

(i) *Recenter*: transforming samples in order to have zero-mean;

(ii) *Standardize*: transforming recentered samples in order to have unit-variance;

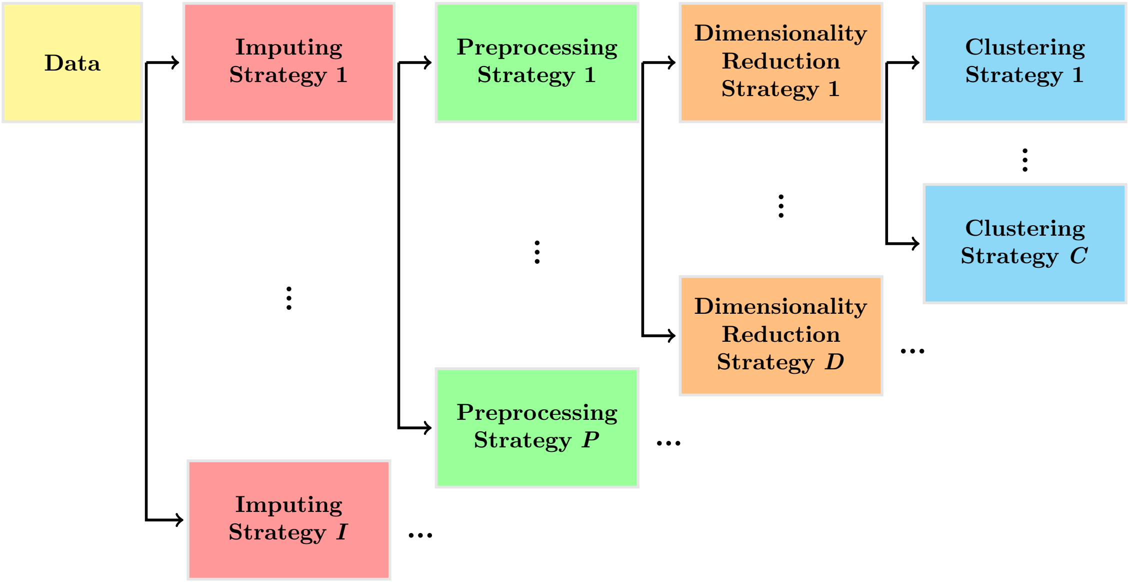

Figure 1: A schematic representation of the `Adenine` workflow. The list of building blocks available for each step is summarized in Table 1. The final number of pipelines generated is $I \times P \times D \times C$.

(iii) *Normalize*: scaling samples in order to have ℓ^p unitary norm (with $p = 1$ or 2);

(iv) *MinMax*: scaling features to a given range, this transformation can be expressed as

$$X_{MinMax} = \frac{X - min(X)}{max(X) - min(X)} \cdot \Big[max(X) - min(X)\Big] + min(X)$$

where *min* and *max* denote, respectively, feature-wise minimum and maximum values operators.

Step 2: Dimensionality reduction. Data exploration of high dimensional dataset can be very tricky. Visualizing samples in high dimension is much less intuitive than representing them in two or three-dimensional scatterplots. Moreover, it is often possible to *decrease* the dimensionality of the problem estimating, by means of different strategies, a low-dimensional embedding in which the data lie. To accomplish this task, `Adenine` offers a set of linear and nonlinear dimensionality reduction and manifold learning algorithms, see Table 1 for the full list of available strategies and the correspondent references.

Step 3: Unsupervised clustering. Cluster analysis, when needed, is the last step of our pipelines. Clustering techniques in `Adenine` are either centroid-based and non centroid-based. For both classes of methods, `Adenine` offers strategies ad heuristics to automatically estimate the best parameter that yields the most suitable cluster separation. The optimal parameter selection of centroid-based algorithms follows the B-fold cross-validation strategy presented in Algorithm 1, where $\mathcal{S}(X, y)$ is defined as the mean silhouette coefficient (Rousseeuw, 1987) for all input samples. The tuning parameter for the affinity propagation technique (Frey and Dueck, 2007) is the so-called *preference* and it defines the number of discovered clusters k. For k-means (Bishop, 2006) the tuning parameter is directly the *number of clusters*, while the mean shift strategy (Comaniciu and Meer, 2002) has an implicit clusters discovery. For hierarchical (Friedman et al., 2001) and spectral clustering (Shi and Malik, 2000) no automatic number of clusters discovery is offered. However graphical aids to evaluate the performance with fixed k are generated as, respectively, dendrogram tree and eigenvalues of the Laplacian of the affinity matrix plot.

Algorithm 1 Automatic discovery of the optimal clustering parameter.

```
for clustering parameter $k$ in $k_1 \dots k_K$ do
    for cross-validation split $b$ in $1 \dots B$ do
        $X_b^{tr}, X_b^{vld} \leftarrow$ $b$-th training, validation set
        $\hat{m} \leftarrow$ fit model on $X_b^{tr}$
        $\hat{y} \leftarrow$ predict labels of $X_b^{vld}$ according to $\hat{m}$
        $s_b \leftarrow$ evaluate silhouette score $\mathcal{S}(X_b^{vld}, \hat{y})$
    end for
    $\bar{S}_k = \frac{1}{B} \sum_{i=1}^{B} s_i$
end for
$k_{opt} = \arg\max_k (\bar{S}_k)$
```

In order to perform exploratory analysis of large datasets, `Adenine` takes advantage of different parallel computing paradigms. `Adenine` pipelines are designed to be independent from each other, therefore they all run in parallel as separate `Python` processes on different cores. Moreover, since `Adenine` makes large use of `numpy` and `scipy`, it automatically benefits from their bindings with optimized linear algebra libraries (such as OpenBLAS[1], or Intel® MKL).

3. Usage Example

Comparing pipelines performance on toy dataset.

4. Experiments and results

To assess the quality of the obtained results, we tested `Adenine` on a set of synthetic and real dataset.

{parla qui dei test synth} {TGCA}

[1] http://www.openblas.net/

Table 1: Pipelines building blocks and their references (not specified when the definition is given in Section 2).

Step	**Algorithms**	**Ref.**
Imputing	mean	
	median	
	most frequent	
	KNN	(Troyanskaya et al., 2001)
Preprocessing	recentering	
	standardize	
	normalize	
	min-max	
Dimensionality reduction	PCA	(Jolliffe, 2002)
	incremental PCA	(Ross et al., 2008)
	randomized PCA	(Halko et al., 2011)
	kernel PCA	(Schölkopf et al., 1997)
	isomap	(Tenenbaum et al., 2000)
	locally linear embedding	(Roweis and Saul, 2000)
	spectral embedding	(Ng et al., 2002)
	multidimensional scaling	(Borg and Groenen, 2005)
	t-distributed stochastic neighbor embedding (t-SNE)	(Van der Maaten and Hinton, 2008)
Clustering	k-means	(Bishop, 2006)
	affinity propagation	(Frey and Dueck, 2007)
	mean shift	(Comaniciu and Meer, 2002)
	spectral	(Shi and Malik, 2000)
	hierarchical	(Friedman et al., 2001)

References

Christopher M Bishop. Pattern recognition. *Machine Learning*, 2006.

Ingwer Borg and Patrick JF Groenen. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media, 2005.

Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619, 2002.

Janez Demšar, Tomaž Curk, Aleš Erjavec, Črt Gorup, Tomaž Hočevar, Mitar Milutinovič, Martin Možina, Matija Polajnar, Marko Toplak, Anže Starič, et al. Orange: data mining toolbox in python. *The Journal of Machine Learning Research*, 14(1):2349–2353, 2013.

Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002.

Joshua M Lewis, Virginia R De Sa, and Laurens Van Der Maaten. Divvy: fast and intuitive exploratory data analysis. *The Journal of Machine Learning Research*, 14(1):3159–3163, 2013.

Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008.

Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Kernel principal component analysis. In *Artificial Neural NetworksICANN'97*, pages 583–588. Springer, 1997.

Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):520–525, 2001.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.