Journal of Machine Learning Research X (2016) X-XX Submitted X/XX; Published XX/XX

ADENINE: A Data ExplorationN 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

ADENINE is a machine learning framework for data exploration. Its goal is to help 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, data 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 plots along with quantitative descriptions of pipeline results. ADENINE is available at `http://slipguru.github.io/adenine/` under FreeBSD license.

Keywords: data exploration, unsupervised learning, dimensionality reduction, clustering,

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 heterogeneous contexts. In these situations, a preliminary data exploration step is not only a 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 software and libraries were released. At a very coarse grain we can group them in two families: GUI-based and command-line applications. Among 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; however, 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 and, as of today, *Orange* lacks in several nonlinear methods such as isomap (Tenenbaum et al., 2000), 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 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 the generation and running of the pipelines composed 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 data analysis concept of *pipeline*. A pipeline is a sequence of the following fundamental steps: (i) missing values imputing, (ii) data preprocessing, (iii) dimensionality reduction and (iv) unsupervised clustering.

For each task, different off-the-shelf algorithms are available (see Table 1).

In order to cope with real-world datasets where entries are often missing, ADENINE offers an improved version of the Imputer class provided by `scikit-learn`. Our extension adds a k-nearest neighbors (k-NN) imputing method to the pre-existent feature-wise *mean*, *median* and *most frequent* value strategies. (Troyanskaya et al., 2001)

Collecting data from heterogeneous sources may imply dealing with features lying in very different numerical ranges. This could have a negative influence on the behavior of dimensionality reduction and clustering techniques. To tackle this issue ADENINE offers different strategies to preprocess data.

Data exploration of high dimensional dataset can be very tricky. However, 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. ADENINE offers a set of linear and nonlinear dimensionality reduction and manifold learning algorithms (see Table 1).

Besides offering a wide range of clustering techniques, ADENINE implements strategies and heuristics to automatically estimate parameters that yield 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 the mean silhouette coefficient (Rousseeuw, 1987) for all input samples.

Clustering parameters for affinity propagation (Frey and Dueck, 2007) and k-means (Bishop, 2006), that are *preference* and *number of clusters*, can be automatically defined. Mean shift (Comaniciu and Meer, 2002) has an implicit cluster discovery. For hierarchical (Friedman et al., 2001) and spectral clustering (Shi and Malik, 2000) no automatic discovery of clustering parameters is offered. However, ADENINE generates graphical aids to evaluate clustering performance with fixed parameters such as, respectively, dendrogram tree and eigenvalues of the Laplacian of the affinity matrix plots.

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
$k_{opt} = \arg\max_k (\bar{S}_k)$

Step	**Algorithm**	**Reference**
Imputing	mean	
	median	
	most frequent	
	k-nearest neighbors	(Troyanskaya et al., 2001)
Preprocessing	recentering	
	standardize	
	normalize	
	min-max	
Dimensionality reduction	principal component analysis (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	(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)

Table 1: Pipeline building blocks available in ADENINE. See ADENINE documentation for a comprehensive description.

(a) (b)

Figure 1: K-means performance after two different nonlinear projections of samples coming from RNA-Seq gene expression dataset available on ADENINE documentation webpage. Data-points colors refer to real classes, while backgrounds are colored according to clustering predictions.

3. Usage Example

In this section we show how to use ADENINE to perform an exploratory analysis on a relatively small dataset. Once ADENINE is installed, all we need to do is to execute the Python script `ade_run.py` specifying as single input argument a configuration file (with `.py` extension) which should look like the snippet below.

```
from adenine.utils import data_source
X, y, feats, classes = data_source.load("custom", "data.csv", "labels.csv")
step1 = {"Normalize": [True, {"norm": "l2"}]} # Preprocessing
step2 = {"KernelPCA": [True, {"kernel": ["rbf"], "n_components": 3, "gamma":
2}], "Isomap": [True, {"n_components": 3}]} # Dimensionality reduction
step3 = {"KMeans": [True, {"n_clusters": ["auto"]}]} # Clustering
```

Each `step` variable refers to a dictionary having the name of the building block as key and a list as value. Each list has a *on**off* trigger in first position followed by a dictionary of keyword arguments for the class implementing the correspondent method. When more than one method is specified in a single step (or a single parameter is passed as list) ADENINE generates the pipelines made by all possible combinations. For a comprehensive description of options specifiable in the configuration file we refer to ADENINE documentation and tutorials: `www.slipguru.unige.it/Software/Adenine`.

The configuration file above generates two pipelines with similar structure. They both have ℓ^2-normalized samples, projected on a three-dimensional space by Gaussian kernel principal component analysis with $\gamma = 2$ (pipeline a) and isomap (pipeline b); on the dimensionality-reduced dataset a k-means clustering with automatic cluster discovery (as in Algorithm 1) is eventually performed. Results of this first step are all stored in a single output folder. Once the analysis are completed, plots and reports can be automatically generated running the Python script `ade_analysis.py` specifying the output folder previously created as single input argument. Figure 1 shows one of several possible comparisons between the two pipelines.

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 Networks—ICANN'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.

John N Weinstein, Eric A Collisson, Gordon B Mills, Kenna R Mills Shaw, Brad A Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, Joshua M Stuart, Cancer Genome Atlas Research Network, et al. The cancer genome atlas pan-cancer analysis project. *Nature genetics*, 45(10):1113–1120, 2013.