Abstract

pyDML is an open-source python library that provides a wide range of distance metric learning algorithms. Distance metric learning can be useful to improve similarity learning algorithms, such as the nearest neighbors classifier, and also has other applications, like dimensionality reduction. The pyDML package currently provides more than 20 algorithms, which can be categorized, according to their purpose, in: dimensionality reduction algorithms, algorithms to improve nearest neighbors or nearest centroids classifiers, information theory based algorithms or kernel based algorithms, among others. In addition, the library also provides some utilities for the visualization of classifier regions, parameter tuning and a stats website with the performance of the implemented algorithms. The package relies on the scipy ecosystem, it is fully compatible with scikit-learn, and is distributed under GPLv3 license. Source code and documentation can be found at https://github.com/jlsuarezdiaz/pyDML.

Keywords: Distance Metric Learning, Classification, Mahalanobis Distance, Dimensionality, Python

1. Introduction

The use of distances in machine learning has been present since its inception, since they provide a similarity measure between the data. Algorithms such as the nearest neighbor classifier (Cover and Hart, 1967) use that similarity measure to label new samples. Traditionally, standard distances, like the euclidean distance, have been used to measure the data similarity. However, a standard distance may not fit our data properly, so the learning results could be non optimal. Finding a distance that brings similar data as close as possible, while moving non similar data away can significantly increase the quality of similarity based learning algorithms. This is the task that distance metric learning carries out.

Distance metric learning (DML) (Suárez et al., 2019; Bellet et al., 2015; Kulis, 2013) is a machine learning discipline with the purpose of learning distances from a dataset. If we focus on Mahalanobis distances, which are expressed as $d_M(x, y) = \sqrt{(x - y)^T M (x - y)}$, where $M$ is a positive semidefinite matrix, learning a distance reduces to learning the matrix $M$. This is equivalent to learning a linear map $L$. In this case, $M = L^T L$ and the learned distance is equivalent to the euclidean distance after applying the transformation $L$ to the data. Therefore, DML algorithms aim at optimizing functions parameterized by a positive
Several DML libraries have been developed in different programming languages. In R, we can find the package dml (Tang et al., 2018), which proposes 11 DML algorithms. However, only 5 algorithms are currently implemented and the package has not exhibited activity for some time. In MATLAB, the DistLearn\(^1\) toolkit provides links to several DML implementations, many of them corresponding to the original authors. However, many of the links are currently broken and the algorithms are not presented in a unified framework. In Python, the metric-learn library (de Vazelhes et al., 2019) provides 9 different DML algorithms, mostly oriented towards weak supervised learning, with the exception of a few classical supervised DML algorithms.

In this paper, we present the pyDML library, a Python package that gathers a wide variety of DML algorithms. The following sections describe the main features of the software, some instructions for installation and usage, several quality standards to which the project subscribes, and finally we expose our plans on future functionalities to be included in the project.

2. Software Description

The pyDML library currently contains around 20 (mostly supervised) algorithms and distances that can be used to prepare a dataset for a subsequent similarity-based learning. These algorithms, and some of their main features, are shown in Table 1.

### Table 1: Current algorithms/distances available in the pyDML library.

<table>
<thead>
<tr>
<th>Algorithm/distance</th>
<th>Supervised</th>
<th>Dimensionality reduction</th>
<th>Oriented to improve...</th>
<th>Information theory based</th>
<th>Kernel version</th>
<th>Learns ... (L or M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euclidean</td>
<td>✓</td>
<td>✓</td>
<td>Just a static distance</td>
<td>✓</td>
<td>✓</td>
<td>Identity</td>
</tr>
<tr>
<td>Covariance</td>
<td>✓</td>
<td>✓</td>
<td>Just a static distance</td>
<td>✓</td>
<td>✓</td>
<td>M</td>
</tr>
<tr>
<td>PCA [Jolliffe, 2002]</td>
<td>✓</td>
<td>✓</td>
<td>Non-specific</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
</tr>
<tr>
<td>LDA [Fukunaga, 1990]</td>
<td>✓</td>
<td>✓</td>
<td>Non-specific</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
</tr>
<tr>
<td>LLDA [Fukunaga, 2007]</td>
<td>✓</td>
<td>✓</td>
<td>Non-specific</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
</tr>
<tr>
<td>ANMM [Wang and Zhang, 2007]</td>
<td>✓</td>
<td>✓</td>
<td>k-NN</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
</tr>
<tr>
<td>LMNN [Weinberger and Saul, 2009]</td>
<td>✓</td>
<td>✓</td>
<td>k-NN</td>
<td>✓</td>
<td>✓</td>
<td>Both</td>
</tr>
<tr>
<td>NCA [Goldberger et al., 2005]</td>
<td>✓</td>
<td>✓</td>
<td>L-NN</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
</tr>
<tr>
<td>NCMML [Stenslau et al., 2013]</td>
<td>✓</td>
<td>✓</td>
<td>Nearest class mean</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
</tr>
<tr>
<td>NCMC [Henschke et al., 2012]</td>
<td>✓</td>
<td>✓</td>
<td>Generalized NCM</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
</tr>
<tr>
<td>ITML [Davis et al., 2001]</td>
<td>✓</td>
<td>✓</td>
<td>Non-specific</td>
<td>✓</td>
<td>✓</td>
<td>M</td>
</tr>
<tr>
<td>DMLMJ [Nguyen et al., 2017]</td>
<td>✓</td>
<td>✓</td>
<td>k-NN</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
</tr>
<tr>
<td>DMLMJ - Non-specific</td>
<td>✓</td>
<td>✓</td>
<td>Non-specific</td>
<td>✓</td>
<td>✓</td>
<td>M</td>
</tr>
<tr>
<td>MCMLM [Gibson and Kowes, 2006]</td>
<td>✓</td>
<td>✓</td>
<td>Non-specific</td>
<td>✓</td>
<td>✓</td>
<td>M</td>
</tr>
<tr>
<td>LSI / MMC [Xiong et al., 2003]</td>
<td>✓</td>
<td>✓</td>
<td>Non-specific</td>
<td>✓</td>
<td>✓</td>
<td>M</td>
</tr>
<tr>
<td>DML-eig [Ying and Li, 2012]</td>
<td>✓</td>
<td>✓</td>
<td>Non-specific</td>
<td>✓</td>
<td>✓</td>
<td>M</td>
</tr>
<tr>
<td>LDML [Goldman et al., 2009]</td>
<td>✓</td>
<td>✓</td>
<td>Non-specific</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
</tr>
<tr>
<td>GMML [Zadeh et al., 2016]</td>
<td>✓</td>
<td>✓</td>
<td>Non-specific</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
</tr>
<tr>
<td>KDA [Mika et al., 1999]</td>
<td>✓</td>
<td>✓</td>
<td>Non-specific</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
</tr>
<tr>
<td>KLLDA [Fukunaga, 2007]</td>
<td>✓</td>
<td>✓</td>
<td>Non-specific</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
</tr>
<tr>
<td>KANMM [Wang and Zhang, 2007]</td>
<td>✓</td>
<td>✓</td>
<td>k-NN</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
</tr>
<tr>
<td>KLMNN [torresani and Lee, 2007]</td>
<td>✓</td>
<td>✓</td>
<td>k-NN</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
</tr>
<tr>
<td>KDMLMJ [Nguyen et al., 2017]</td>
<td>✓</td>
<td>✓</td>
<td>k-NN</td>
<td>✓</td>
<td>✓</td>
<td>L</td>
</tr>
</tbody>
</table>

Python, the chosen programming language, is widely used in machine learning, and has several libraries specialized in this field. The main one is Scikit-Learn (Pedregosa et al., 2011), an efficient open-source library for machine learning, which relies on the Scipy\(^2\).

---

2. https://www.scipy.org
ecosystem, which contains numerical calculus libraries, such as NumPy, data processing libraries, such as Pandas, or data visualization libraries, such as Matplotlib. Python has been also chosen because until now it does not have an extensive library with supervised DML algorithms. pyDML tries to fill this gap, providing numerous supervised DML algorithms, both classic algorithms and new proposals.

The design followed for the development of the algorithms has preserved the structure of the algorithms of the Scikit-Learn library. In particular, the DML algorithms are included in the group of transformation algorithms, where the transformation consists in applying the learned linear map to the samples. Therefore, the implemented algorithms inherit from a template class DMLAlgorithm, which in turn inherits from the sklearn.base.TransformerMixin\(^3\) class of the Scikit-Learn toolkit. This hierarchy allows the DML algorithms to be treated as black-box transformers, which facilitates their handling and pipelining with other Scikit-Learn algorithms. The DMLAlgorithm class provides the inherited methods fit(X, y) and transform(X), to learn the distance and apply it to the data, following the Scikit-Learn syntax, as well as the specific methods metric(), transformer() and metadata() that allow us to access the learned metric matrix, the learned linear map or several metadata generated during the learning process, respectively.

It is important to emphasize that these algorithms include different hyperparameters that can be modified to improve the performance or to change the conditions of the learned distances. To this end the package includes tune functions, which allow the parameters of the DML algorithms to be easily estimated with cross validation, using the success rate of a k-neighbors classifier or some of the metadata of the algorithms as validation metrics. A detailed description of all hyperparameters for each algorithm can be found in the pyDML’s full documentation\(^4\).

The pyDML library also incorporates graphical tools for the representation and evaluation of the learned distances, which use the Matplotlib library internally. These tools allow labeled data to be represented, along with the regions determined by any Scikit-Learn classifier, including distance-based classifiers, for which several functionalities are provided to easily represent the effect of different distances.

## 3. Installation and Usage

The pyDML library can be installed through PyPI (Python package index), using the command pip install pyDML. It is also possible to download or clone the repository directly from GitHub. In such a case, the installation of the software package can be done by running the setup script available in the root directory, using the command python setup.py install. Once installed, we can access all DML algorithms, and the additional functionalities, by importing the desired class within the dml module.

---

As already mentioned, the way DML algorithms are used is similar to the Scikit-Learn transformers. Figure 1 shows a basic example. More detailed examples of all the possibilities offered by pyDML can be found in the documentation¹.

```python
>>> import numpy as np  # NumPy library
>>> from sklearn.datasets import load_iris  # Iris dataset
>>> from dml import NCA  # Loading DML algorithm

>>> # Loading dataset
>>> iris = load_iris()
>>> X = iris['data']
>>> y = iris['target']

>>> nca = NCA()  # DML construction
>>> nca.fit(X, y)  # Fitting algorithm

>>> # We can look at the algorithm metadata
>>> meta = nca.metadata()
>>> meta
{'final_expectance': 0.95771240234375,
 'initial_expectance': 0.8380491129557291,
 'num_iters': 3}

>>> # After fitting it
>>> M = nca.metric()
>>> M
array([[ 1.1909, 0.5129, -2.1581, -2.0146],
       [ 0.5129, 1.5812, -2.1457, -2.1071],
       [-2.1581, -2.1457, 6.4688, 5.8628],
       [-2.0146, -2.1071, 5.8628, 6.8327]])

>>> # Equivalently, we can see the learned linear map.
>>> L = nca.transformer()
>>> L
array([[ 0.7796, -0.0191, -0.3586, -0.2399],
       [-0.0444, 1.0074, -0.2993, -0.2581],
       [-0.6074, -0.5728, 2.1609, 1.3521],
       [-0.4606, -0.4875, 1.2573, 2.2091]])

>>> # Finally, we can obtain the transformed data,
>>> Lx = nca.transform()  # Transforming training set.
>>> Lx[:5, :]
array([[ 3.3590, 2.8288, -1.8073, -1.8538],
       [ 3.2126, 2.3339, -1.3993, -1.5179],
       [ 3.0887, 2.5743, -1.6085, -1.6490],
       [ 2.9410, 2.4181, -1.0583, -1.3027],
       [ 3.2791, 2.9340, -1.8038, -1.8565]])
```

Figure 1: Use of distance metric learning algorithms in pyDML.

### 4. Quality Standards

The project code follows the PEP8 style standards for Python code. Continuous integration is performed, using the Travis CI service, to ensure back-compatibility and integrate code in a simple way. The project adheres to Semantic Versioning and uses the Keep a Changelog standards to make it easier to users to see the changes between each version. A thorough documentation is provided using sphinx and numpydoc, and is hosted in the Read the Docs platform. Finally, a stats website is also provided⁶, where the performance of the implemented algorithms is evaluated under different conditions (Suárez et al., 2019). Those algorithms available in other DML libraries are also compared to each other in this website.

### 5. Conclusions and Future Work

In this paper, we presented a new Python library that integrates a wide range of distance metric learning algorithms, with additional functionalities such as visualization or parameter estimation. The pyDML library is fully compatible with Scikit-Learn and is distributed under GPLv3 license.

---

². https://jlsuarezdiaz.github.io/software/pyDML/stats/index.html#
As future work we plan to extend the library by adding more recent algorithms, and also algorithms oriented to problems beyond standard classification, like ordinal classification (Nguyen et al., 2018), imbalanced classification (Wang et al., 2018) or non-standard problems (Charte et al. 2019). We will also explore new ways of learning distances beyond the Mahalanobis approach, such as deep metric learning (Yi et al. 2014).

Acknowledgments

Our work has been supported by the research project TIN2017-89517-P and by a research scholarship (FPU18/05989), given to the author Juan Luis Suárez by the Spanish Ministry of Science, Innovation and Universities.

References


