Journal of Machine Learning Research 21 (2020) 1-7 Submitted 10/19; Revised 3/20; Published 4/20

pyDML: A Python Library for Distance Metric Learning

Juan Luis Suárez JLSUAREZDIAZ@UGR.ES
Salvador García SALVAGL@DECSAI.UGR.ES
Francisco Herrera HERRERA@DECSAI.UGR.ES
DaSCI, Andalusian Research Institute in Data Science and Computational Intelligence
University of Granada, Granada, Spain

Editor: Andreas Mueller

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 semidefinite (also called metric) matrix M, or by a linear map L (Weinberger and Saul, 2009; Ying and Li, 2012).

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.

Algorithm/distance	Supervised	Dimensionality reduction	Oriented to improve...	Information theory based	Kernel version	Learns ... (L or M)
Euclidean	✗	✗	Just a static distance	✗	✗	Identity
Covariance	✗	✗	Just a static distance	✗	✗	M
PCA (Jolliffe, 2002)	✗	✓	Non-specific	✗	✗	L
LDA (Fisher, 1936)	✓	✓	Non-specific	✗	✗	L
LLDA (Sugiyama, 2007)	✓	✓	Non-specific	✗	✗	L
ANMM (Wang and Zhang, 2007)	✓	✓	k-NN	✗	✗	L
LMNN (Weinberger and Saul, 2009)	✓	✓	k-NN	✗	✗	Both
NCA (Goldberger et al., 2005)	✓	✓	1-NN	✗	✗	L
NCMML (Mensink et al., 2013)	✓	✓	Nearest class mean	✗	✗	L
NCMC (Mensink et al., 2013)	✓	✓	Generalized NCM	✗	✗	L
ITML (Davis et al., 2007)	Weak	✗	Non-specific	✓	✗	M
DMLMJ (Nguyen et al., 2017)	✓	✓	k-NN	✓	✗	L
MCML (Globerson and Roweis, 2006)	✓	✗	Non-specific	✓	✗	M
LSI / MMC (Xing et al., 2003)	Weak	✗	Non-specific	✗	✗	M
DML-eig (Ying and Li, 2012)	Weak	✗	Non-specific	✗	✗	M
LDML (Guillaumin et al., 2009)	✓	✗	Non-specific	✗	✗	M
GMML (Zadeh et al., 2016)	Weak	✗	Non-specific	✗	✗	M
KDA (Mika et al., 1999)	✓	✓	Non-specific	✗	✓	L
KLLDA (Sugiyama, 2007)	✓	✓	Non-specific	✗	✓	L
KANMM (Wang and Zhang, 2007)	✓	✓	k-NN	✗	✓	L
KLMNN (Torresani and Lee, 2007)	✓	✓	k-NN	✗	✓	L
KDMLMJ (Nguyen et al., 2017)	✓	✓	k-NN	✓	✓	L

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

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]

1. `https://www.cs.cmu.edu/~liuy/distlearn.htm`
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 `DML_Algorithm`, 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 `DML_Algorithm` 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.

3. `http://scikit-learn.org/stable/modules/generated/sklearn.base.TransformerMixin.html#sklearn.base.TransformerMixin`
4. `https://pydml.readthedocs.io/`

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 posibilities offered by `pyDML` can be found in the documentation[5].

```
>>> 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
>>> # after fitting it
>>> meta = nca.metadata()
>>> meta
{'final_expectance': 0.95771240234375,
 'initial_expectance': 0.8380491129557291,
 'num_iters': 3}
>>> # We can see the metric the algorithm has learned.
>>> 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,
>>> # or transform new 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 provided[6], 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.

5. `https://pydml.readthedocs.io/en/latest/examples.html`
6. `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

A. Bellet, A. Habrard, and M. Sebban. *Metric Learning.* Morgan & Claypool, 2015.

D. Charte, F. Charte, S. García, and F. Herrera. A snapshot on nonstandard supervised learning problems: taxonomy, relationships, problem transformations and algorithm adaptations. *Progress in Artificial Intelligence*, 8(1):1–14, 2019.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Proceedings of the 24th International Conference on Machine learning*, pages 209–216. ACM, 2007.

W. de Vazelhes, C. Carey, Y. Tang, N. Vauquier, and A. Bellet. metric-learn: Metric learning algorithms in python. *arXiv preprint arXiv:1908.04710*, 2019.

R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

A. Globerson and S. T. Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems*, pages 451–458, 2006.

J. Goldberger, G. E. Hinton, S. T. Roweis, and R. R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems*, pages 513–520, 2005.

M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? metric learning approaches for face identification. In *2009 IEEE 12th International Conference on Computer Vision*, pages 498–505. IEEE, 2009.

I. Jolliffe. *Principal Component Analysis.* Springer Series in Statistics. Springer, 2002.

B. Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4): 287–364, 2013.

T. Mensink, J. Verbeek, F. Perronnin, and G. Csurka. Distance-based image classification: Generalizing to new classes at near-zero cost. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(11):2624–2637, 2013.

S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K.-R. Mullers. Fisher discriminant analysis with kernels. In *Neural networks for signal processing IX, 1999. Proceedings of the 1999 IEEE signal processing society workshop.*, pages 41–48. IEEE, 1999.

B. Nguyen, C. Morell, and B. De Baets. Supervised distance metric learning through maximization of the jeffrey divergence. *Pattern Recognition*, 64:215–225, 2017.

B. Nguyen, C. Morell, and B. De Baets. Distance metric learning for ordinal classification based on triplet constraints. *Knowledge-Based Systems*, 142:17–28, 2018.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

J. L. Suárez, S. García, and F. Herrera. A tutorial on distance metric learning: Mathematical foundations, algorithms and experiments. *arXiv preprint arXiv:1812.05944*, 2019.

M. Sugiyama. Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. *Journal of Machine Learning Research*, 8(May):1027–1061, 2007.

Y. Tang, T. Gao, and N. Xiao. dml: Distance metric learning in r. *Journal of Open Source Software*, 3(30):1036, 2018.

L. Torresani and K.-C. Lee. Large margin component analysis. In *Advances in Neural Information Processing Systems*, pages 1385–1392, 2007.

F. Wang and C. Zhang. Feature extraction by maximizing the average neighborhood margin. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

N. Wang, X. Zhao, Y. Jiang, and Y. Gao. Iterative metric learning for imbalance data classification. In *International Joint Conferences on Artificial Intelligence*, pages 2805–2811, 2018.

K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10(Feb):207–244, 2009.

E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng. Distance metric learning with application to clustering with side-information. In *Advances in Neural Information Processing Systems*, pages 521–528, 2003.

D. Yi, Z. Lei, S. Liao, and S. Z. Li. Deep metric learning for person re-identification. In *2014 IEEE 22nd International Conference on Pattern Recognition*, pages 34–39. IEEE, 2014.

Y. Ying and P. Li. Distance metric learning with eigenvalue optimization. *Journal of Machine Learning Research*, 13(Jan):1–26, 2012.

P. Zadeh, R. Hosseini, and S. Sra. Geometric mean metric learning. In *International Conference on Machine Learning*, pages 2464–2471, 2016.