The Journal of Open Source Software

LightTwinSVM: A Simple and Fast Implementation of Standard Twin Support Vector Machine Classifier

Amir M. Mir[1] **and Jalal A. Nasiri**[1]

1 Iranian Research Institute for Information Science and Technology (IranDoc), Tehran, Iran

DOI: 10.21105/joss.01252

Software

- Review
- Repository
- Archive

Submitted: 05 January 2019
Published: 31 March 2019

Abstract

This paper presents the `LightTwinSVM` program and its features. It is a simple and fast implementation of twin support vector machine algorithm (TwinSVM). Numerical experiments on benchmark datasets show the effectiveness of the `LightTwinSVM` program in terms of accuracy. This program can be used by researchers, students, and practitioners to solve classifications tasks.

Introduction

Classification is a widely-used learning method in machine learning. The task of classification consists of training samples for which the class labels are available. On the basis of such training samples, a classifier learns a rule for predicting an unseen sample (Shalev-Shwartz & Ben-David, 2014). To do a classification task, many algorithms have been proposed in machine learning literature such as Artificial Neural Network (ANN), Support Vector Machine (SVM), K-nearest neighbors (KNN), and Decision Trees. Among these classification algorithms, SVM classifier has relatively better prediction accuracy and generalization (Kotsiantis, Zaharakis, & Pintelas, 2007). The main idea of SVM is to find the optimal separating hyperplane with the largest margin between the two classes. Figure 1 shows the geometric illustration of SVM classifier.

Over the past decade, researchers have proposed new classifiers based on the SVM (Nayak, Naik, & Behera, 2015). Of these extensions of SVM, the twin support vector machine (TwinSVM) (Jayadeva, Khemchandani, & Chandra, 2007) has received more attention from scholars in the field of SVM research. This may be due to the novel idea of TwinSVM which is doing classification using two non-parallel hyperplanes. Each of which is as close as possible to samples of its own class and far from samples of the other class. To show the central idea of TwinSVM graphically, Figure 2 shows the geometric illustration of TwinSVM classifier.

For SVMs, there exist several stable software packages and implementations such as `LIBSVM` (C.-C. Chang & Lin, 2011) and `LIBLINEAR` (Fan, Chang, Hsieh, Wang, & Lin, 2008). These packages were used to implement SVM in `scikit-learn` (Pedregosa et al., 2011) which is a widely-used machine learning package for Python programming language. To solve a classification problem with the SVM algorithm, one can use `scikit-learn` with only a few lines of code in Python.

Even though TwinSVM is a popular classification algorithm in the field of SVM research, to the best of our knowledge, there exists no free and reliable implementation with a user guide on the internet. This motivated us to develop `LightTwinSVM` program to

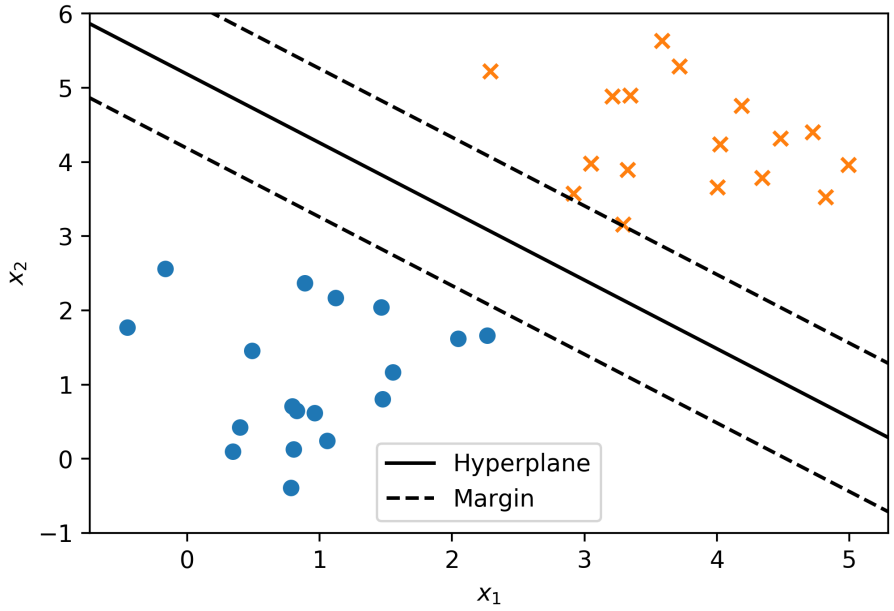

Figure 1: Geometric illustration of SVM classifier.

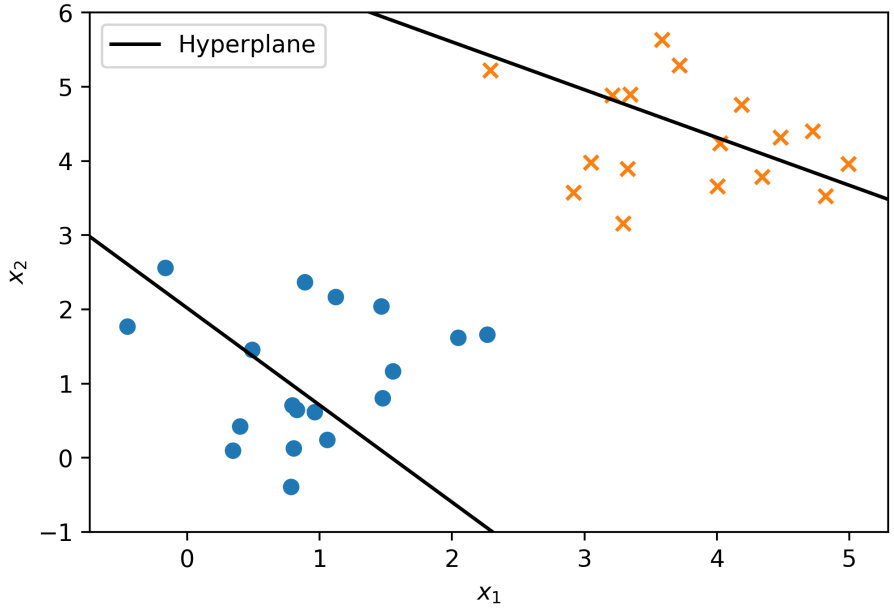

Figure 2: Geometric illustration of TwinSVM classifier.

help researchers, practitioners, and students build their own classifier on the basis of `LightTwinSVM`. Moreover, this program can be used to solve classification problems. In the next section, we present `LightTwinSVM`, its features, and compare it with `scikit-learn`'s SVM.

LightTwinSVM

The `LightTwinSVM` program is a simple and fast implementation of the TwinSVM classifier. It is mostly written in Python and its main design goals are simplicity and speed. Also, this program is free, open source, and licensed under the terms of GNU GPL v3[1]. `LightTwinSVM` is built on top of `NumPy` (Walt, Colbert, & Varoquaux, 2011), `scikit-learn` (Pedregosa et al., 2011), and `pandas` (McKinney, 2011).

`LightTwinSVM` program can be used by both researchers in the field of SVM research and by students in courses on pattern recognition and machine learning. Moreover, this software can be applied to a wide variety of research applications such as text classification, image or video recognition, medical diagnosis, and bioinformatics. For example, `Light TwinSVM` was used for the numerical experiments in our previous research paper (Mir & Nasiri, 2018).

The main features of the `LightTwinSVM` program are the following:

- To make its usage simple, a **command-line application** was created to help users solve classification tasks step-by-step.
- Since speed is one of the design goals, the **clipDCD optimization algorithm** (Peng, Chen, & Kong, 2014) is employed which is a simple and fast external optimizer. It was improved and implemented in C++.
- In order to solve linear or non-linear classification problems, both **linear** and **RBF kernels** are supported.
- Multi-class classification problems can be solved using either **One-vs-One** or **One-vs-All** scheme.
- The One-vs-One classifier is **scikit-learn compatible**. Therefore, `scikit-learn` tools such as GridSearchCV and cross_val_score can be employed.
- To evaluate the performance of the classifier, **K-fold cross-validation** and **train/test split** are supported.
- The optimal values of hyper-parameters can be found with **grid search**.
- **CSV** and **LIBSVM** formats are supported for importing datasets.
- Detailed classification results are saved in a spreadsheet file so that results can be analyzed and interpreted.

The source code of `LightTwinSVM`, its installation guide and usage example can be found at https://github.com/mir-am/LightTwinSVM.

Numerical Experiments

In this section, we conducted experiments to show the efficiency of `LightTwinSVM` program, and compared it with the implementation of SVM in `scikit-learn` on the UCI[2] benchmark datasets. To evaluate the classifiers' performance, 5-fold cross-validation is used. For both standard SVM and TwinSVM, the penalty parameter C was selected

[1] https://opensource.org/licenses/GPL-3.0
[2] http://archive.ics.uci.edu/ml/datasets.html

from the set $\{2^i \mid i = -10, -9, \ldots, 5\}$. Moreover, the RBF kernel was used and its parameter γ was chosen over the range $\{2^i \mid i = -15, -14, \ldots, 5\}$. Since the classification performance of standard SVM and TwinSVM depends heavily on the optimal choice of hyper-parameters, grid search is used to find the optimal values of hyper-parameters.

To analyze the classification performance of `LightTwinSVM` and `scikit-learn`'s SVM, the results on benchmark datasets are summarized in Table 1.

Table 1: The accuracy comparison between `LightTwinSVM` and `scikit-learn`'s SVM

Datasets	LightTwinSVM	scikit-learn's SVM	Difference in Accuracy
Pima-Indian	**78.91±3.73**	78.26±2.62	0.65
Australian	**87.25±2.27**	86.81±3.22	0.44
Haberman	76.12±4.79	**76.80±2.68**	-0.68
Cleveland	**85.14±5.45**	84.82±4.04	0.32
Sonar	**84.62±4.89**	64.42±6.81	20.2
Heart-Statlog	**85.56±2.96**	85.19±2.62	0.37
Hepatitis	**86.45±5.16**	83.23±3.55	3.22
WDBC	**98.24±1.36**	98.07±0.85	0.17
Spectf	**81.68±5.35**	79.78±0.19	1.9
Titanic	81.93±2.59	**82.27±1.83**	-0.34
Mean Accuracy	**84.59**	81.94	2.65

From the Table 1, it can be seen that `LightTwinSVM` outperforms `scikit-learn`'s SVM on most datasets. For instance, the accuracy difference in Sonar dataset is as high as 20.2% which is a significant result. However, in consideration of the mean accuracy, one may notice that the difference in accuracy between the two classifiers is not very large. To show whether a significant difference exists, statistical tests are often used in research papers on classification (Demšar, 2006). Due to the limited space, comprehensive experiments with statistical tests are skipped in this paper. In summary, the experiment indicates that the `LightTwinSVM` program can be used for classification tasks and it may produce a better prediction accuracy.

Acknowledgments

This research work was carried out at the machine learning lab of IranDoc Institution. We would like to thank the director of IranDoc Institution for providing us with research facilities.

References

Chang, C.-C., & Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, *2*(3), 27. doi:10.1145/1961189.1961199

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, *7*(Jan), 1–30. Journal Article.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., & Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of machine learning research*, *9*(Aug), 1871–1874.

Jayadeva, Khemchandani, R., & Chandra, S. (2007). Twin support vector machines for pattern classification. *IEEE Transactions on pattern analysis and machine intelligence*, *29*(5). Journal Article. doi:10.1109/TPAMI.2007.1068

Kotsiantis, S. B., Zaharakis, I., & Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, *160*, 3–24.

McKinney, W. (2011). Pandas: A foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, 1–9.

Mir, A., & Nasiri, J. A. (2018). KNN-based least squares twin support vector machine for pattern classification. *Applied Intelligence*, *48*(12), 4551–4564. doi:10.1007/s10489-018-1225-z

Nayak, J., Naik, B., & Behera, H. (2015). A comprehensive survey on support vector machine in data mining tasks: Applications and challenges. *International Journal of Database Theory and Application*, *8*(1), 169–186. Journal Article.

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

Peng, X., Chen, D., & Kong, L. (2014). A clipping dual coordinate descent algorithm for solving support vector machines. *Knowledge-Based Systems*, *71*, 266–278. Journal Article. doi:10.1016/j.knosys.2014.08.005

Shalev-Shwartz, S., & Ben-David, S. (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press. doi:10.1017/CBO9781107298019

Walt, S. van der, Colbert, S. C., & Varoquaux, G. (2011). The numpy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, *13*(2), 22–30. Journal Article. doi:10.1109/MCSE.2011.37