LIBTwinSVM

A Library for Twin Support Vector Machines

Amir Mir

Software Engineering Research Group
EEMCS Faculty

Delft University of Technology

November 13, 2019

Contents

Bio

- Jan. 2019: Received a Master's degree in **Computer Engineering** (Specialization in **Artificial Intelligence**).

Bio

- Jan. 2019: Received a Master's degree in **Computer Engineering** (Specialization in **Artificial Intelligence**).
- Worked as a Research Assistant at IranDoc Institution from Jul. 2017 until Sep. 2019.

Bio

- Jan. 2019: Received a Master's degree in **Computer Engineering** (Specialization in **Artificial Intelligence**).
- Worked as a Research Assistant at IranDoc Institution from Jul. 2017 until Sep. 2019.
- **Research interests**: Machine Learning and its application in Software Engineering.

Background

Support Vector Machine (SVM)

- Introduced by Vapnik and Cortes [1] in 1995.
- Finds an optimal separating hyperplane.

[1] C. Cortes and V. Vapnik (1995). "Support-vector networks". In: *Machine learning* 20.3, pp. 273–297. ISSN: 0885-6125

Background

Support Vector Machine (SVM)

- Introduced by Vapnik and Cortes [1] in 1995.
- Finds an optimal separating hyperplane.

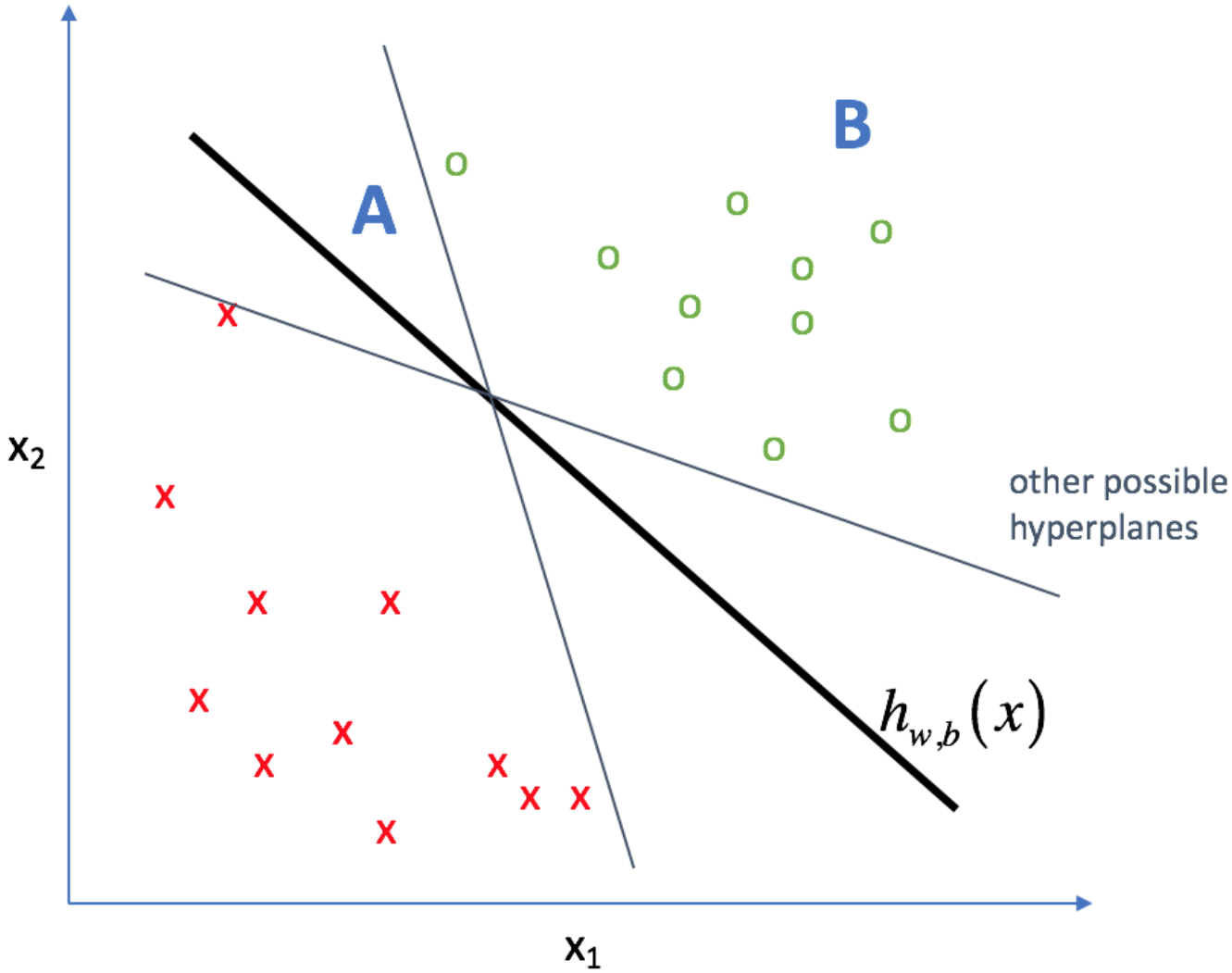
SVM's optimization problem

$$\begin{aligned} &\min_{w} \frac{1}{2}\|w\|^2 \\ &\text{s.t. } y_i(w^T x_i + b) \geq 1, \forall i \end{aligned} \tag{1}$$

[1] C. Cortes and V. Vapnik (1995). "Support-vector networks". In: *Machine learning* 20.3, pp. 273–297. ISSN: 0885-6125

Background

The Illustration of SVM's basic idea



Background

Twin Support Vector Machine (TSVM)

- Inspired by SVM, TSVM[2] was proposed in 2007.
- Two non-parallel hyperplanes.

[2] R Khemchandani, S. Chandra, et al. (2007). "Twin support vector machines for pattern classification". In: *IEEE Transactions on pattern analysis and machine intelligence* 29.5, pp. 905–910

Background

Twin Support Vector Machine (TSVM)

- Inspired by SVM, TSVM[2] was proposed in 2007.
- Two non-parallel hyperplanes.

TSVM's optimization problems

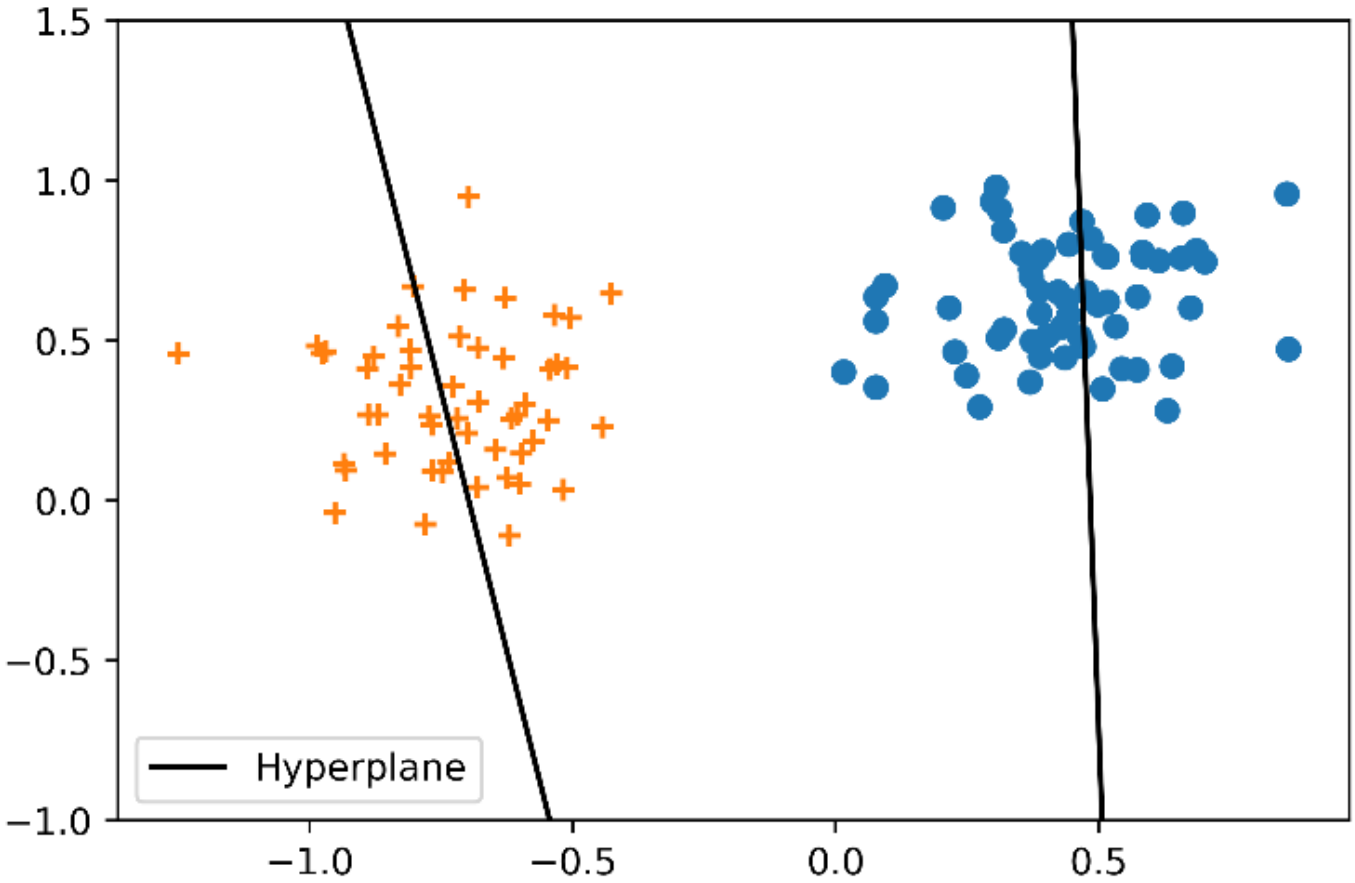
$$\begin{aligned} \underset{\mathbf{w}_1, b_1}{min} \quad & \frac{1}{2}\|\mathbf{A}\mathbf{w}_1 + \mathbf{e}_1 b_1\|^2 + c_1 \mathbf{e}_2^T \xi \\ \text{s.t.} \quad & -(\mathbf{B}\mathbf{w}_1 + \mathbf{e}_2 b_1) + \xi \geq \mathbf{e}_2 \ , \xi \geq 0 \end{aligned} \tag{2}$$

$$\begin{aligned} \underset{\mathbf{w}_2, b_2}{min} \quad & \frac{1}{2}\|\mathbf{B}\mathbf{w}_2 + \mathbf{e}_2 b_2\|^2 + c_2 \mathbf{e}_1^T \eta \\ \text{s.t.} \quad & (\mathbf{A}\mathbf{w}_2 + \mathbf{e}_1 b_2) + \eta \geq \mathbf{e}_1 \ , \eta \geq 0 \end{aligned} \tag{3}$$

[2] R Khemchandani, S. Chandra, et al. (2007). "Twin support vector machines for pattern classification". In: *IEEE Transactions on pattern analysis and machine intelligence* 29.5, pp. 905–910

Background

The Illustration of TSVM's basic idea



Existing Software Packages

- **SVMs**:

[3] A. M. Mir and J. A. Nasiri (2019). "LightTwinSVM: A Simple and Fast Implementation of Standard Twin Support Vector Machine Classifier". In: *Journal of Open Source Software* 4, p. 1252

Existing Software Packages

- **SVMs**:
  - **LIBLINEAR:** supports linear SVMs.

[3] A. M. Mir and J. A. Nasiri (2019). "LightTwinSVM: A Simple and Fast Implementation of Standard Twin Support Vector Machine Classifier". In: *Journal of Open Source Software* 4, p. 1252

Existing Software Packages

- **SVMs**:
  - **LIBLINEAR:** supports linear SVMs.
  - **LIBSVM:** supports non-linear SVMs.

[3] A. M. Mir and J. A. Nasiri (2019). "LightTwinSVM: A Simple and Fast Implementation of Standard Twin Support Vector Machine Classifier". In: *Journal of Open Source Software* 4, p. 1252

Existing Software Packages

- **SVMs**:
  - **LIBLINEAR:** supports linear SVMs.
  - **LIBSVM:** supports non-linear SVMs.
  - **ThunderSVM:** provides GPU implementation of SVMs.

[3] A. M. Mir and J. A. Nasiri (2019). "LightTwinSVM: A Simple and Fast Implementation of Standard Twin Support Vector Machine Classifier". In: *Journal of Open Source Software* 4, p. 1252

Existing Software Packages

- **SVMs**:
  - **LIBLINEAR:** supports linear SVMs.
  - **LIBSVM:** supports non-linear SVMs.
  - **ThunderSVM:** provides GPU implementation of SVMs.
- **TSVMs**:
  - **LightTwinSVM:**[3] supports linear and non-linear TSVM classifier.

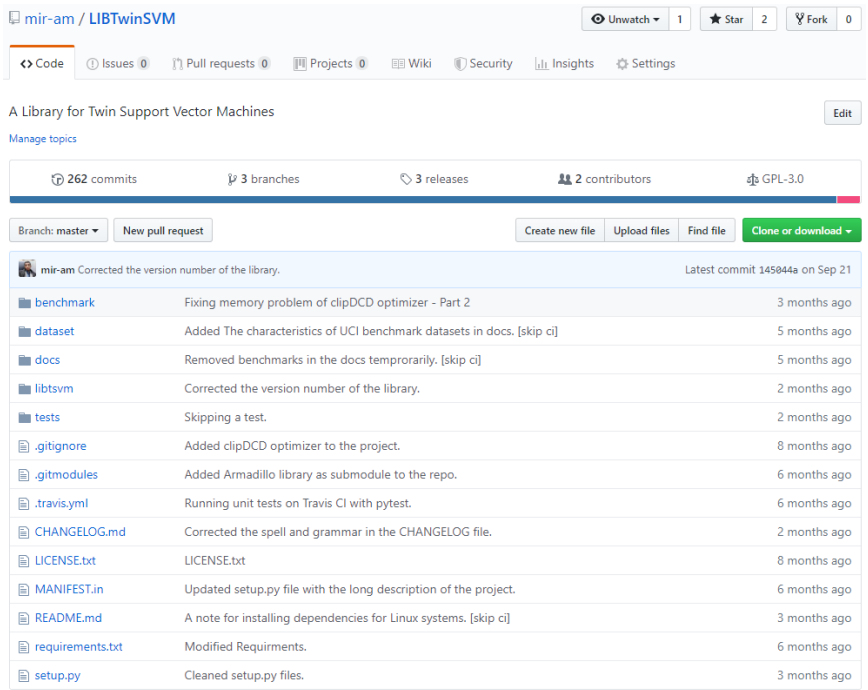
[3] A. M. Mir and J. A. Nasiri (2019). "LightTwinSVM: A Simple and Fast Implementation of Standard Twin Support Vector Machine Classifier". In: *Journal of Open Source Software* 4, p. 1252

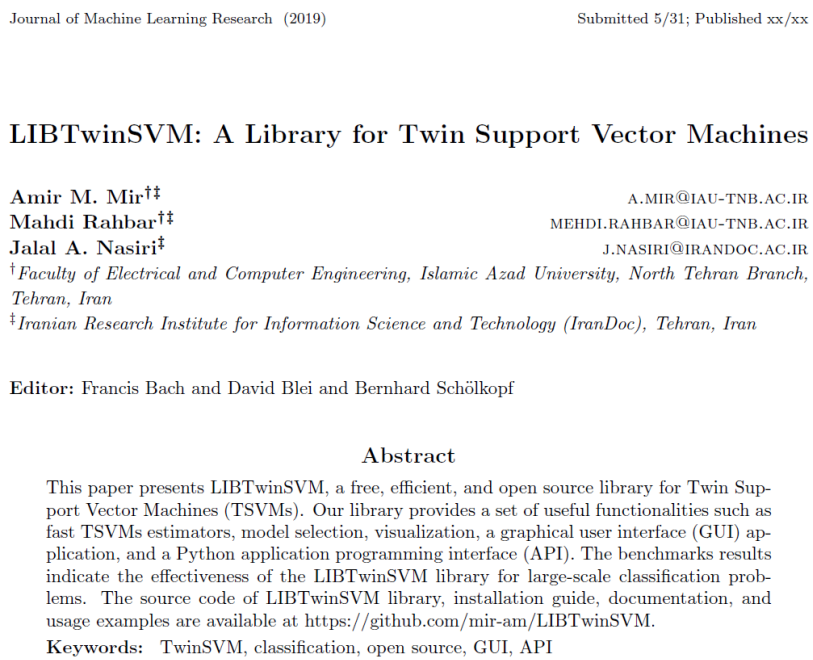
An Intro to LIBTwinSVM Library

- A free, efficient and open-source library for Twin Support Vector Machines (TSVMs).
- It can be used for solving classification tasks.

An Intro to LIBTwinSVM Library

- A free, efficient and open-source library for Twin Support Vector Machines (TSVMs).
- It can be used for solving classification tasks.





Journal of Machine Learning Research (2019) Submitted 5/31; Published xx/xx

LIBTwinSVM: A Library for Twin Support Vector Machines

Amir M. Mir[†‡] A.MIR@IAU-TNB.AC.IR
Mahdi Rahbar[†‡] MEHDI.RAHBAR@IAU-TNB.AC.IR
Jalal A. Nasiri[‡] J.NASIRI@IRANDOC.AC.IR
[†] *Faculty of Electrical and Computer Engineering, Islamic Azad University, North Tehran Branch, Tehran, Iran*
[‡] *Iranian Research Institute for Information Science and Technology (IranDoc), Tehran, Iran*

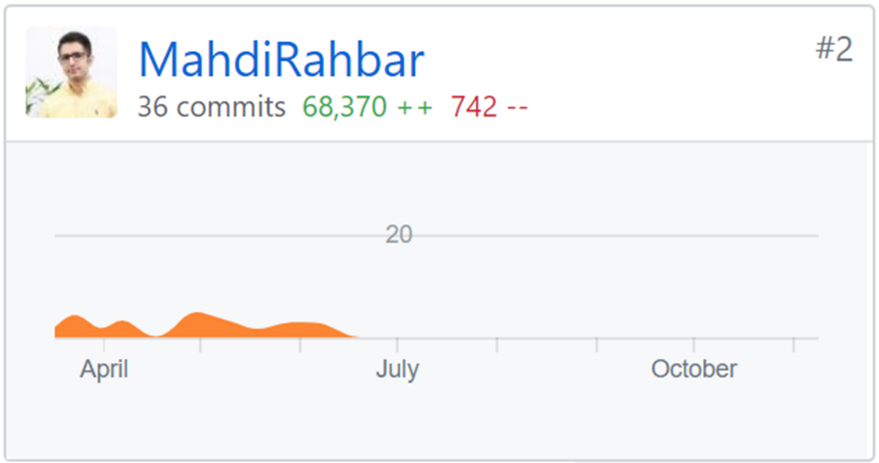
Editor: Francis Bach and David Blei and Bernhard Schölkopf

Abstract

This paper presents LIBTwinSVM, a free, efficient, and open source library for Twin Support Vector Machines (TSVMs). Our library provides a set of useful functionalities such as fast TSVMs estimators, model selection, visualization, a graphical user interface (GUI) application, and a Python application programming interface (API). The benchmarks results indicate the effectiveness of the LIBTwinSVM library for large-scale classification problems. The source code of LIBTwinSVM library, installation guide, documentation, and usage examples are available at https://github.com/mir-am/LIBTwinSVM.

Keywords: TwinSVM, classification, open source, GUI, API

Contributors of the LIBTwinSVM project



Dr. Jalal A. Nasiri

PhD in Computer Engineering; Tarbiat Modares University

- Assistant professor; Information Science Research Department
- j.nasiri@irandoc.ac.ir

The main goals of the LIBTwinSVM project

- Simplicity and ease of use.

The main goals of the LIBTwinSVM project

- Simplicity and ease of use.
- A Python application programming interface (API).

The main goals of the LIBTwinSVM project

- Simplicity and ease of use.
- A Python application programming interface (API).
- Fast running time.

The main goals of the LIBTwinSVM project

- Simplicity and ease of use.
- A Python application programming interface (API).
- Fast running time.
- Providing quite a number of useful features.

Main features of the LIBTwinSVM Library

- A simple and user-friendly Graphical User Interface (GUI).

[4] M. A. Kumar and M. Gopal (2009). "Least squares twin support vector machines for pattern classification". In: *Expert Systems with Applications* 36.4, pp. 7535–7543

[5] X. Peng, D. Chen, and L. Kong (2014). "A clipping dual coordinate descent algorithm for solving support vector machines". In: *Knowledge-Based Systems* 71, pp. 266–278

Main features of the LIBTwinSVM Library

- A simple and user-friendly Graphical User Interface (GUI).
- Supports both **standard TSVM** and its **Least Squares extension**[4].

[4] M. A. Kumar and M. Gopal (2009). "Least squares twin support vector machines for pattern classification". In: *Expert Systems with Applications* 36.4, pp. 7535–7543

[5] X. Peng, D. Chen, and L. Kong (2014). "A clipping dual coordinate descent algorithm for solving support vector machines". In: *Knowledge-Based Systems* 71, pp. 266–278

Main features of the LIBTwinSVM Library

- A simple and user-friendly Graphical User Interface (GUI).
- Supports both **standard TSVM** and its **Least Squares extension**[4].
- A **fast optimizer** (ClipDCD[5]) in C++.

[4] M. A. Kumar and M. Gopal (2009). "Least squares twin support vector machines for pattern classification". In: *Expert Systems with Applications* 36.4, pp. 7535–7543

[5] X. Peng, D. Chen, and L. Kong (2014). "A clipping dual coordinate descent algorithm for solving support vector machines". In: *Knowledge-Based Systems* 71, pp. 266–278

Main features of the LIBTwinSVM Library

- A simple and user-friendly Graphical User Interface (GUI).
- Supports both **standard TSVM** and its **Least Squares extension**[4].
- A **fast optimizer** (ClipDCD[5]) in C++.
- Supports **Linear**, Radial Basis Function (**RBF**), and **Rectangular** kernels.

[4] M. A. Kumar and M. Gopal (2009). "Least squares twin support vector machines for pattern classification". In: *Expert Systems with Applications* 36.4, pp. 7535–7543

[5] X. Peng, D. Chen, and L. Kong (2014). "A clipping dual coordinate descent algorithm for solving support vector machines". In: *Knowledge-Based Systems* 71, pp. 266–278

Main features of the LIBTwinSVM Library

- A simple and user-friendly Graphical User Interface (GUI).
- Supports both **standard TSVM** and its **Least Squares extension**[4].
- A **fast optimizer** (ClipDCD[5]) in C++.
- Supports **Linear**, Radial Basis Function (**RBF**), and **Rectangular** kernels.
- Supports popular **multi-class classification** approaches such as One-vs-All and One-vs-One.

[4] M. A. Kumar and M. Gopal (2009). "Least squares twin support vector machines for pattern classification". In: *Expert Systems with Applications* 36.4, pp. 7535–7543

[5] X. Peng, D. Chen, and L. Kong (2014). "A clipping dual coordinate descent algorithm for solving support vector machines". In: *Knowledge-Based Systems* 71, pp. 266–278

Main features of the LIBTwinSVM Library

- A simple and user-friendly Graphical User Interface (GUI).
- Supports both **standard TSVM** and its **Least Squares extension**[4].
- A **fast optimizer** (ClipDCD[5]) in C++.
- Supports **Linear**, Radial Basis Function (**RBF**), and **Rectangular** kernels.
- Supports popular **multi-class classification** approaches such as One-vs-All and One-vs-One.
- **Scikit-learn-compatible** estimators.

[4] M. A. Kumar and M. Gopal (2009). "Least squares twin support vector machines for pattern classification". In: *Expert Systems with Applications* 36.4, pp. 7535–7543

[5] X. Peng, D. Chen, and L. Kong (2014). "A clipping dual coordinate descent algorithm for solving support vector machines". In: *Knowledge-Based Systems* 71, pp. 266–278

Main features of the LIBTwinSVM Library

- A simple and user-friendly Graphical User Interface (GUI).
- Supports both **standard TSVM** and its **Least Squares extension**[4].
- A **fast optimizer** (ClipDCD[5]) in C++.
- Supports **Linear**, Radial Basis Function (**RBF**), and **Rectangular** kernels.
- Supports popular **multi-class classification** approaches such as One-vs-All and One-vs-One.
- **Scikit-learn-compatible** estimators.
- A **visualization tool** to show the decision boundaries of TSVMs.

[4] M. A. Kumar and M. Gopal (2009). "Least squares twin support vector machines for pattern classification". In: *Expert Systems with Applications* 36.4, pp. 7535–7543

[5] X. Peng, D. Chen, and L. Kong (2014). "A clipping dual coordinate descent algorithm for solving support vector machines". In: *Knowledge-Based Systems* 71, pp. 266–278

Main features of the LIBTwinSVM Library

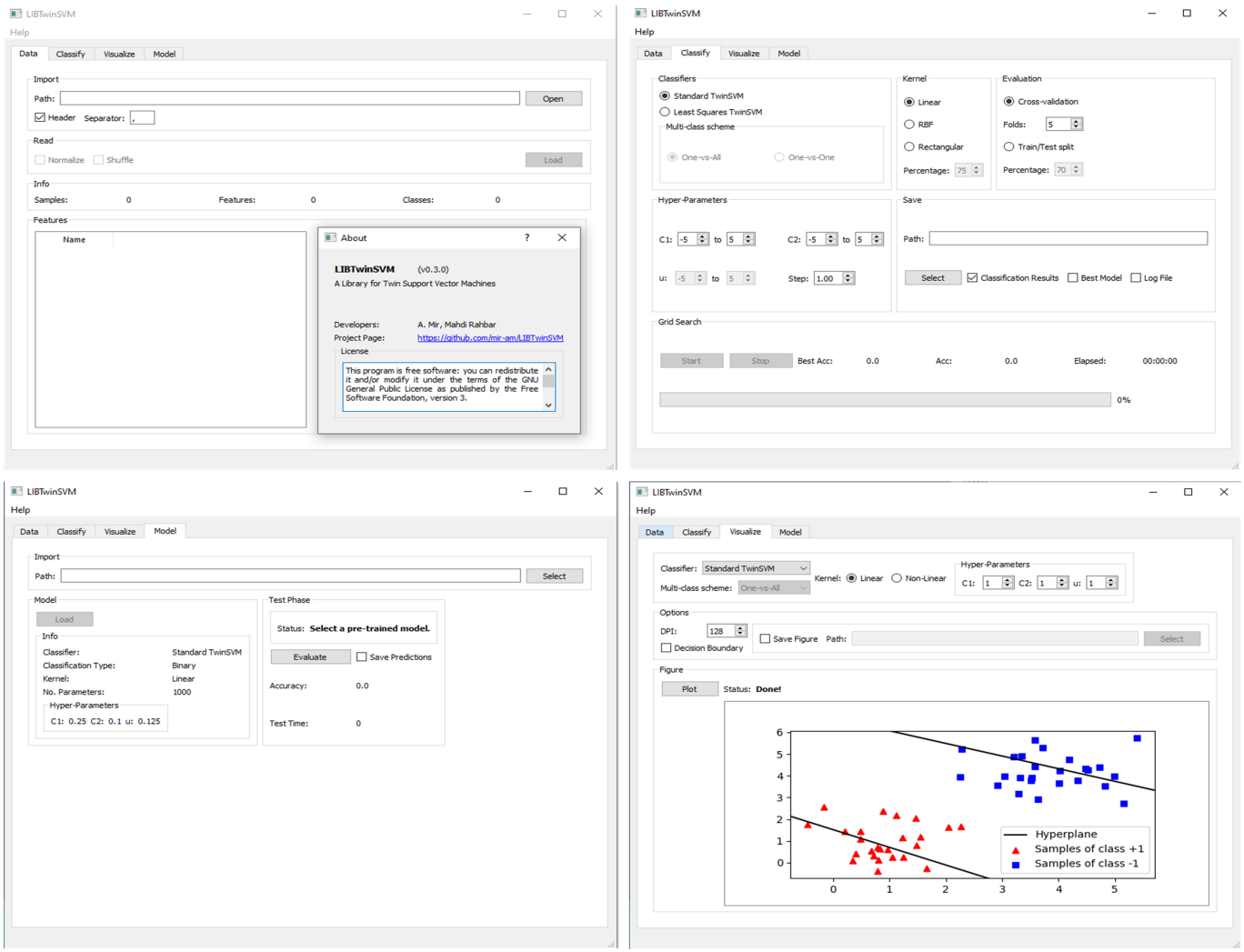
- A simple and user-friendly Graphical User Interface (GUI).
- Supports both **standard TSVM** and its **Least Squares extension**[4].
- A **fast optimizer** (ClipDCD[5]) in C++.
- Supports **Linear**, Radial Basis Function (**RBF**), and **Rectangular** kernels.
- Supports popular **multi-class classification** approaches such as One-vs-All and One-vs-One.
- **Scikit-learn-compatible** estimators.
- A **visualization tool** to show the decision boundaries of TSVMs.
- It can saves the best-fitted model on the disk and also can load pre-trained models.

[4] M. A. Kumar and M. Gopal (2009). "Least squares twin support vector machines for pattern classification". In: *Expert Systems with Applications* 36.4, pp. 7535–7543

[5] X. Peng, D. Chen, and L. Kong (2014). "A clipping dual coordinate descent algorithm for solving support vector machines". In: *Knowledge-Based Systems* 71, pp. 266–278

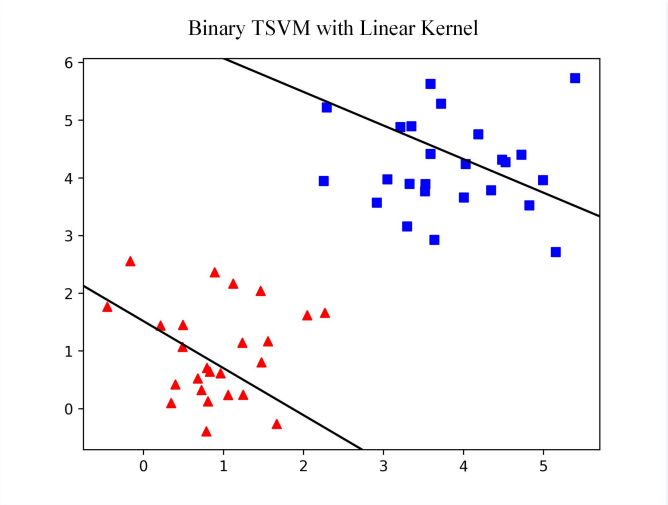
Main features of the LIBTwinSVM Library

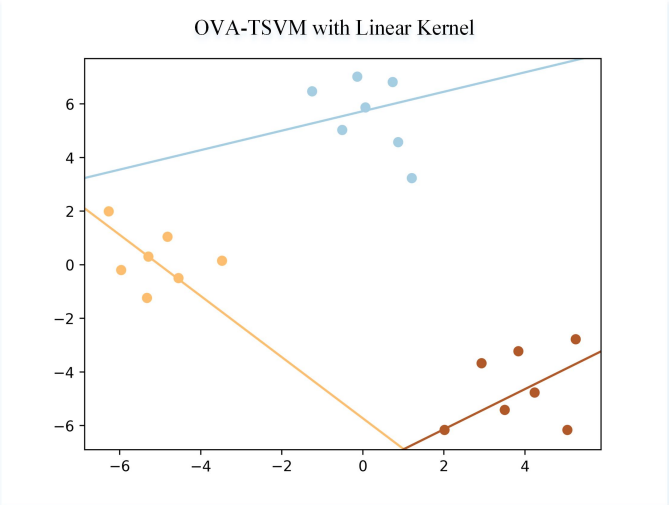
The GUI application

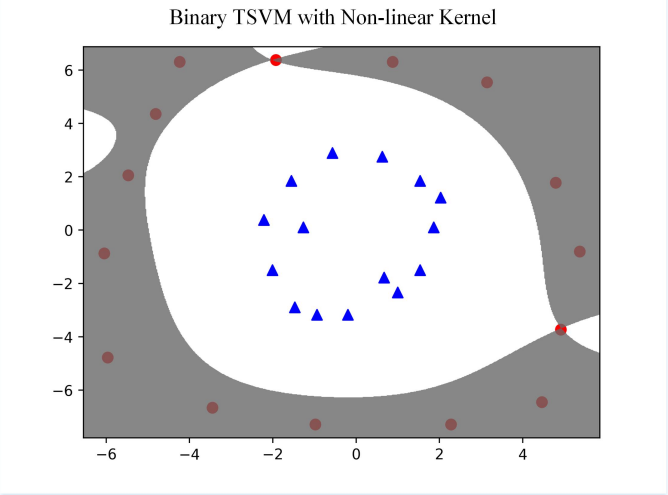


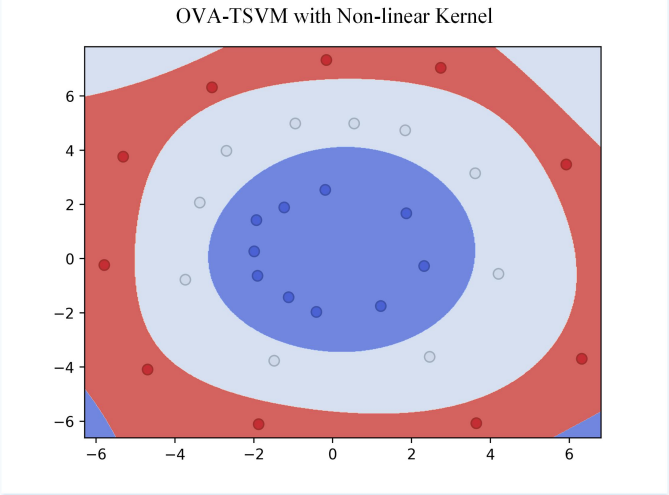
Main features of the LIBTwinSVM Library

The visualization feature



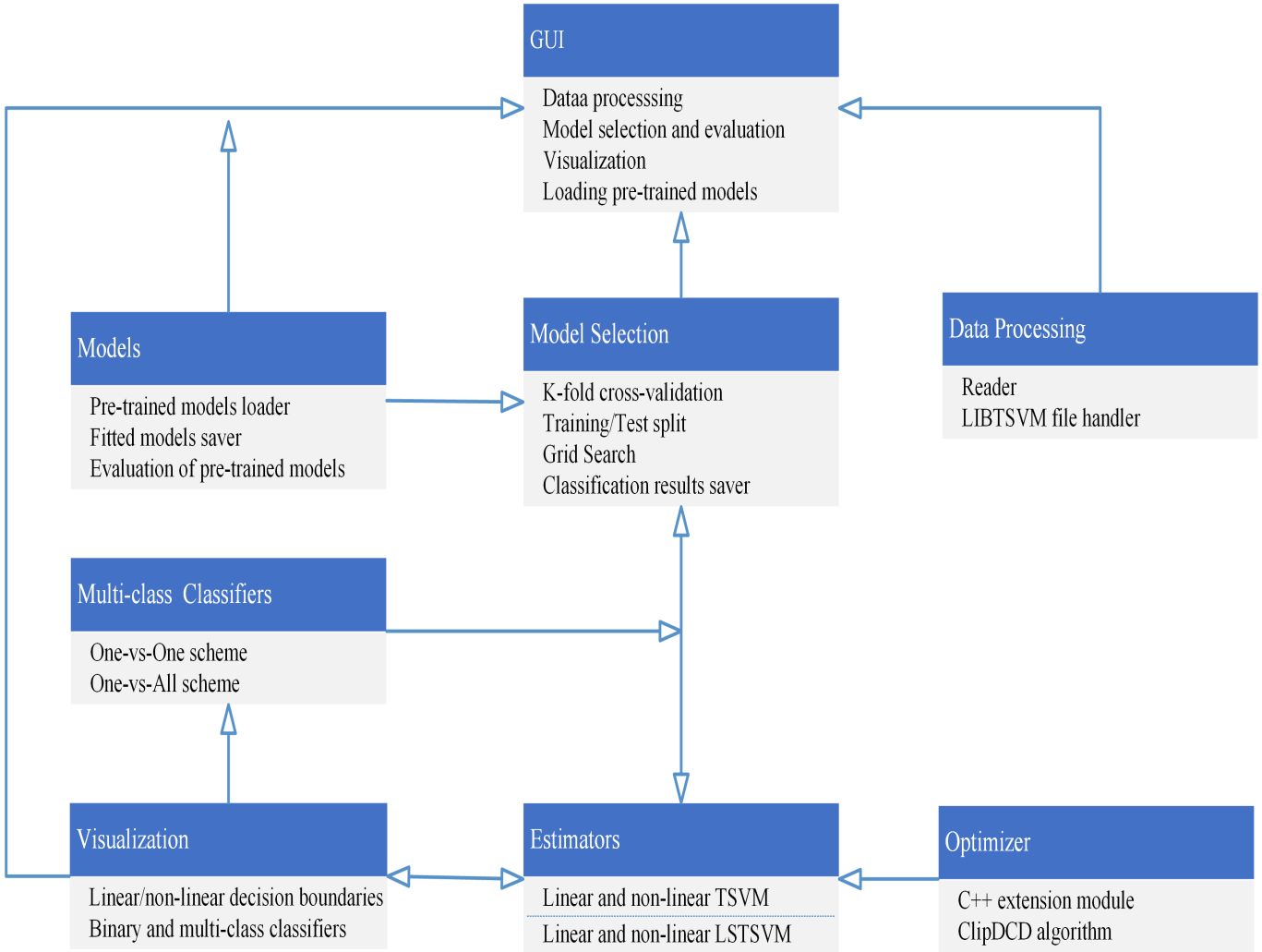






Design

Main components of the LIBTwinSVM library



Design

Core components

- **Estimators:** An interface for the TSVM classifier and its Least Squares extension.

Design

Core components

- **Estimators:** An interface for the TSVM classifier and its Least Squares extension.
- **Multiclass classifiers:** An interface for One-vs-All and One-vs-One approaches.

Design

Core components

- **Estimators:** An interface for the TSVM classifier and its Least Squares extension.
- **Multiclass classifiers:** An interface for One-vs-All and One-vs-One approaches.
- **Model Selection**: K-fold cross-validation, Train/Test split, and grid search

Design

Core components

- **Estimators:** An interface for the TSVM classifier and its Least Squares extension.
- **Multiclass classifiers:** An interface for One-vs-All and One-vs-One approaches.
- **Model Selection**: K-fold cross-validation, Train/Test split, and grid search
- **Optimizer**: ClipDCD algorithm for solving optimization problems of TSVM.

Implementation

Dependencies of the LIBTwinSVM library

Package	Usage
	Python
NumPy	Fast linear algebra operations such as matrix multiplication and inverse
Scikit-learn	For the evaluation and selection of TSVM-based models
Matplotlib	Visualization and geometrical interpretation of TSVMs
PyQT5	A GUI for using the functionalities of the library
Pandas	For reading and processing datasets
XlsxWriter	For saving classification results in an Excel file
Joblib	For saving and loading TwinSVM-based models
numpydoc	API code documentation
	C++
Armadillo	C++ linear algebra library for optimizer component
LAPACK and BLAS	Numerical linear algebra.

Implementation

Optimizer component

- It is the performance-critical part of the library.

[6] https://cvxopt.org/

Implementation

Optimizer component

- It is the performance-critical part of the library.
- We could use CVXOPT[6] package.

[6] https://cvxopt.org/

Implementation

Optimizer component

- It is the performance-critical part of the library.
- We could use CVXOPT[6] package.
- Instead, clipDCD algorithm is implemented in C++.

[6] https://cvxopt.org/

Implementation

Optimizer component

- It is the performance-critical part of the library.
- We could use CVXOPT[6] package.
- Instead, clipDCD algorithm is implemented in C++.
- Cython was used for generating Python extension module.

[6] https://cvxopt.org/

Implementation

Optimizer component

- It is the performance-critical part of the library.
- We could use CVXOPT[6] package.
- Instead, clipDCD algorithm is implemented in C++.
- Cython was used for generating Python extension module.
- The implementation caches major computation with the space cost of $\mathcal{O}(n)$.

[6] https://cvxopt.org/

Implementation

Optimizer component

- It is the performance-critical part of the library.
- We could use CVXOPT[6] package.
- Instead, clipDCD algorithm is implemented in C++.
- Cython was used for generating Python extension module.
- The implementation caches major computation with the space cost of $\mathcal{O}(n)$.
- Memory copies are avoided.

[6] https://cvxopt.org/

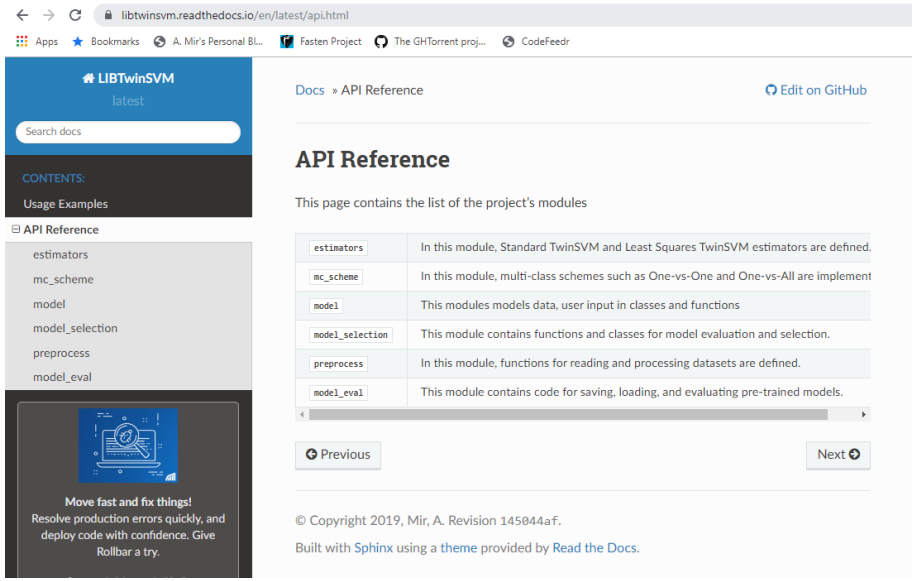
API

- Inspired by the design of Scikit-learn package[7].

[7] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, et al. (2013). "API design for machine learning software: experiences from the scikit-learn project". In: *arXiv preprint arXiv:1309.0238*

API

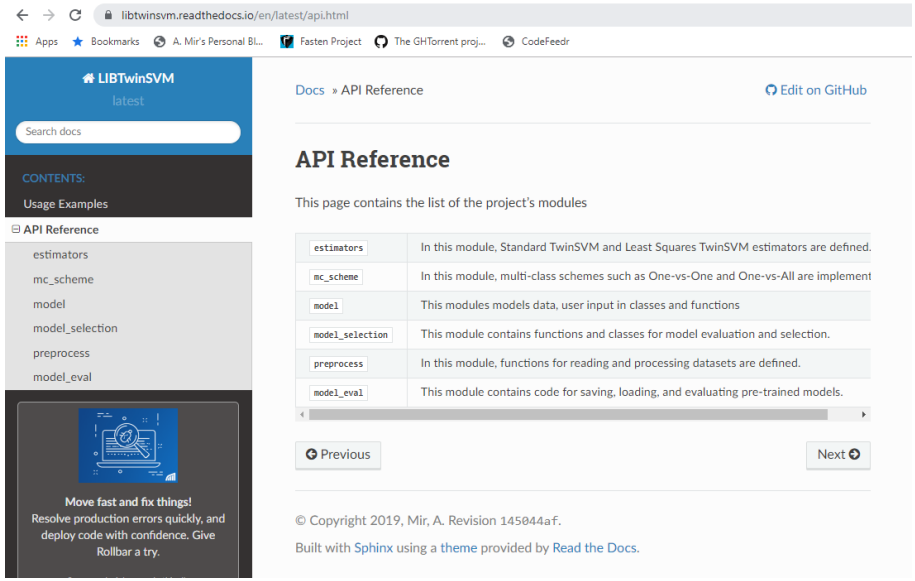
- Inspired by the design of Scikit-learn package[7].
- The interfaces of the estimators provides `fit()` and `predict()`.



[7] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, et al. (2013). "API design for machine learning software: experiences from the scikit-learn project". In: *arXiv preprint arXiv:1309.0238*

API

- Inspired by the design of Scikit-learn package[7].
- The interfaces of the estimators provides `fit()` and `predict()`.
- API reference and usage examples are provided.



[7] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, et al. (2013). "API design for machine learning software: experiences from the scikit-learn project". In: *arXiv preprint arXiv:1309.0238*

API

Usage example

```
from libtsvm.preprocess import DataReader
from libtsvm.estimators import TSVM
from libtsvm.model_selection import Validator

# Step 1: Load your dataset
data_path = '../../dataset/australian.csv'
sep_char = ',' # separtor character of the CSV file
header = True # Whether the dataset has header names.

dataset = DataReader(data_path, sep_char, header)

shuffle_data = True
normalize_data = False

dataset.load_data(shuffle_data, normalize_data)
X, y, file_name = dataset.get_data()

# Step 2: Choose a TSVM-based estimator
kernel = 'linear'
tsvm_clf = TSVM(kernel=kernel)

# Step 3: Evaluate the estimator using train/test split
eval_method = 't_t_split' # Train/Test split
test_set_size = 30 # 30% of samples

val = Validator(X, y, (eval_method, test_set_size), tsvm_clf)
eval_func = val.choose_validator()

# Hyper-parameters of the classifier
h_params =  {'C1': 2**-3, 'C2': 2**-5}

acc, std, full_report = eval_func(h_params)

print("Accuracy: %.2f" % acc)
print(full_report)
```

Benchmarks

Classification performance

Experiment details:

- The datasets were obtained from **UCI**[8] repository.

[8] https://archive.ics.uci.edu/ml/index.php

Benchmarks

Classification performance

Experiment details:

- The datasets were obtained from **UCI**[8] repository.
- All the datasets were normalized in the range $[0, 1]$.

[8] https://archive.ics.uci.edu/ml/index.php

Benchmarks

Classification performance

Experiment details:

- The datasets were obtained from **UCI**[8] repository.
- All the datasets were normalized in the range $[0, 1]$.
- **5-fold cross-validation** was used for the evaluation.

[8] https://archive.ics.uci.edu/ml/index.php

Benchmarks

Classification performance

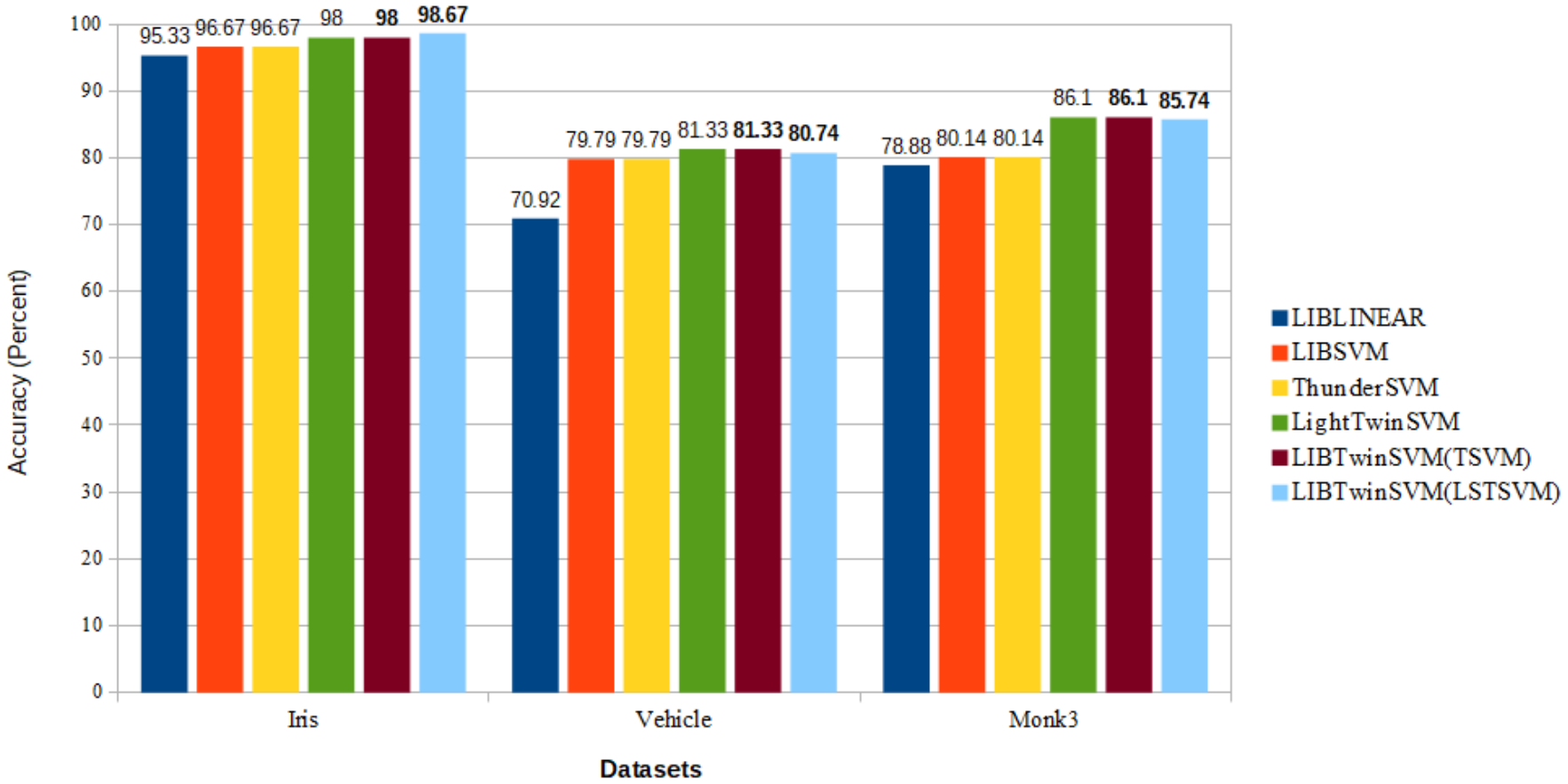
Experiment details:

- The datasets were obtained from **UCI**[8] repository.
- All the datasets were normalized in the range $[0, 1]$.
- **5-fold cross-validation** was used for the evaluation.
- The **grid search** was used to find the optimal value of hyper-parameters.

[8] https://archive.ics.uci.edu/ml/index.php

Benchmarks

Classification performance



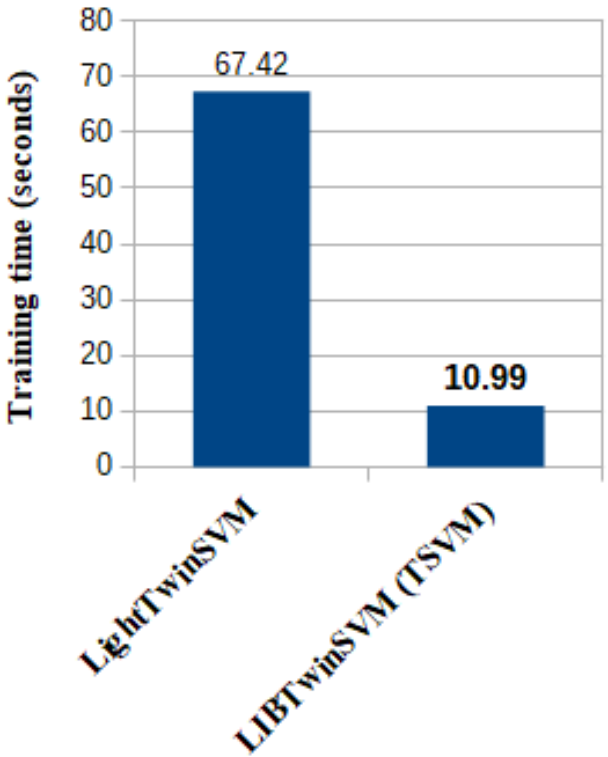
Benchmarks

Speed: Training time

NDC-50K dataset[9]

LightTwinSVM vs. LIBTwinSVM(TSVM)

About 6 times improvement



[9] D. Musicant (1998). "NDC: normally distributed clustered datasets". In: *Computer Sciences Department, University of*

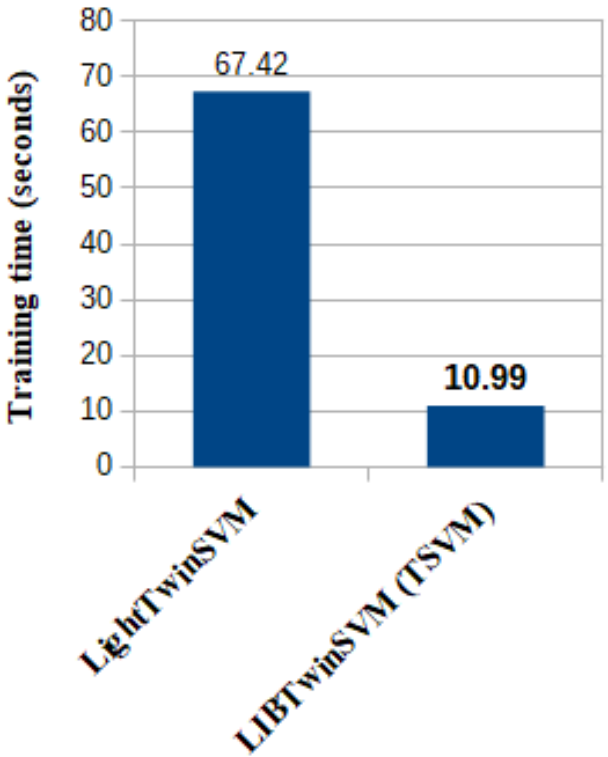
Benchmarks

Speed: Training time

NDC-50K dataset[9]

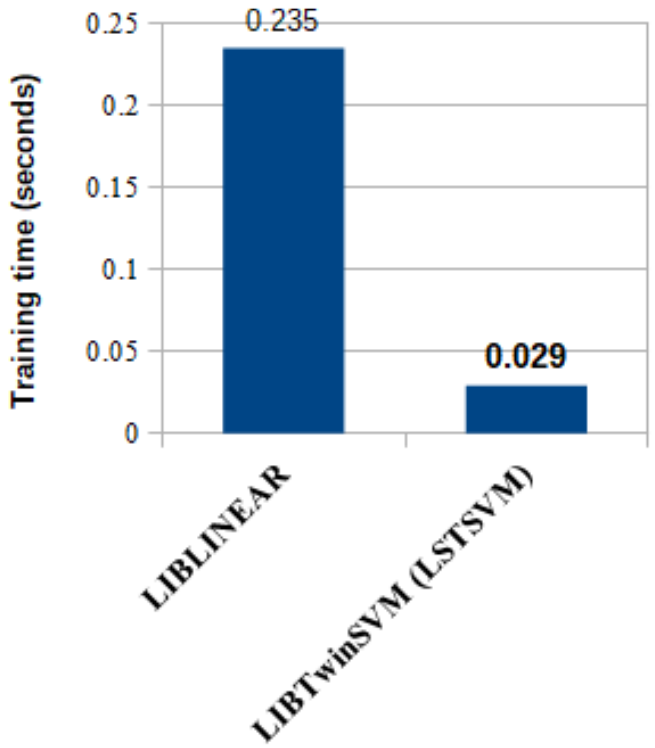
LightTwinSVM vs. LIBTwinSVM(TSVM)

About 6 times improvement



LIBLINEAR vs. LIBTwinSVM(LSTSVM)

About 8 times faster



[9] D. Musicant (1998). "NDC: normally distributed clustered datasets". In: *Computer Sciences Department, University of*

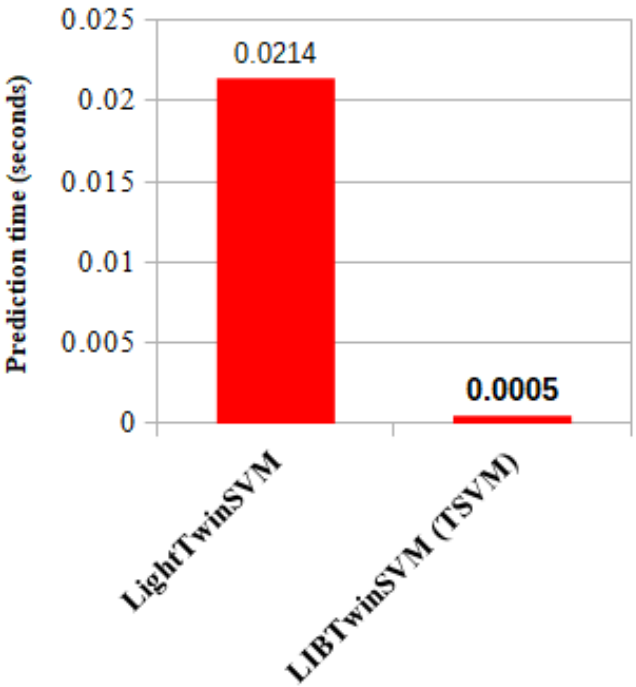
Benchmarks

Speed: Prediction time

NDC-50K dataset

LightTwinSVM vs. LIBTwinSVM(TSVM)

About 42 times improvement



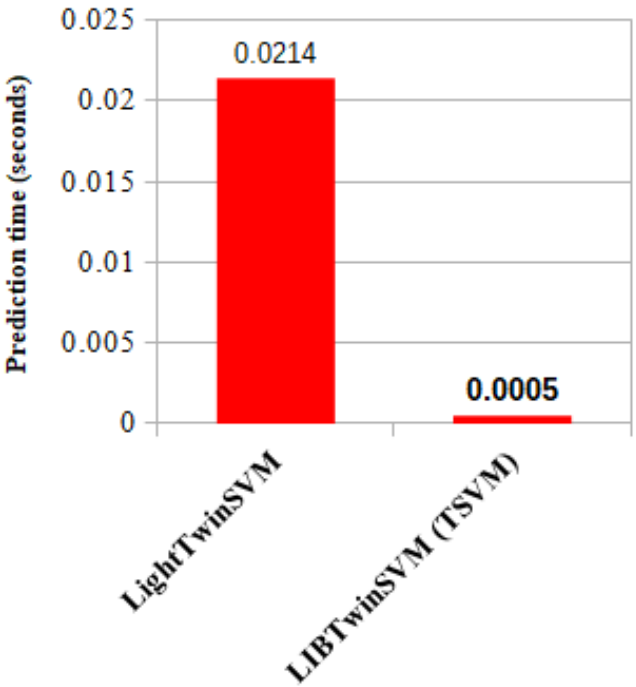
Benchmarks

Speed: Prediction time

NDC-50K dataset

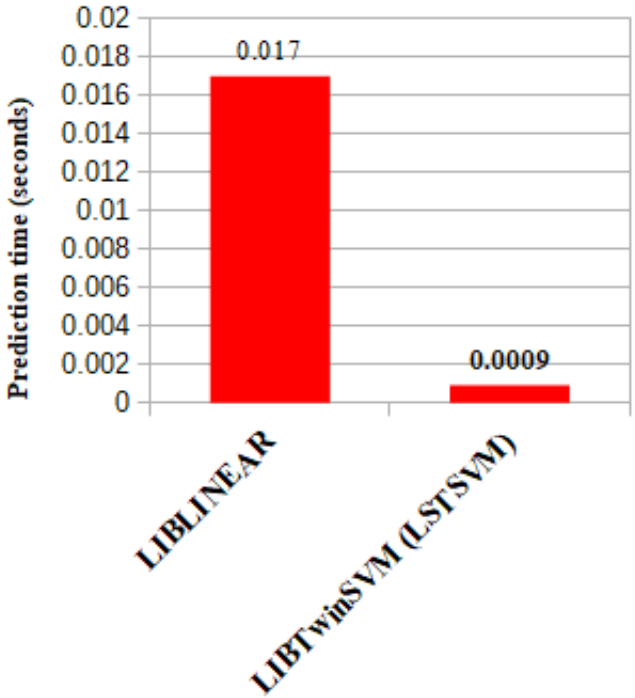
LightTwinSVM vs. LIBTwinSVM(TSVM)

About 42 times improvement



LIBLINEAR vs. LIBTwinSVM(LSTSVM)

About 18 times faster



Benchmarks

Memory consumption

Procedure for measuring the peak memory consumption:

- Loading a dataset.
- Training an estimator using `fit()` method.
- The Linux `time` command for measuring the usage.

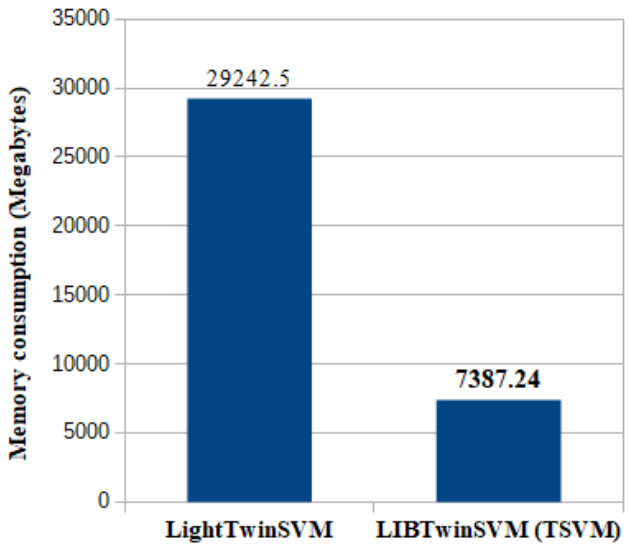
Benchmarks

Memory consumption

NDC-50K dataset

LightTwinSVM vs. LIBTwinSVM(TSVM)

About 4 times lower



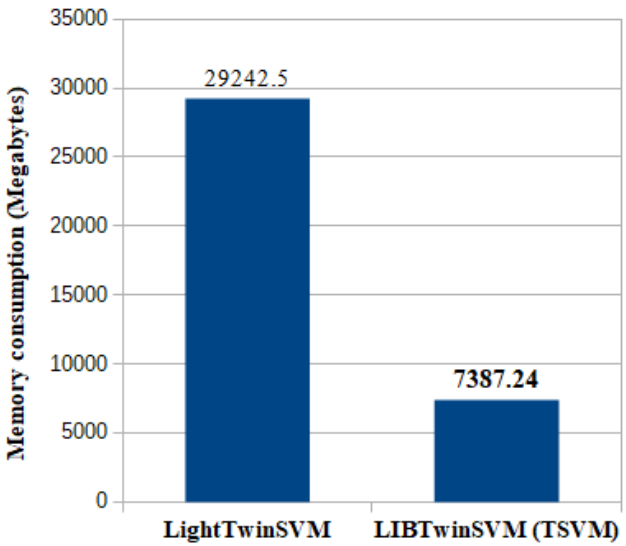
Benchmarks

Memory consumption

NDC-50K dataset

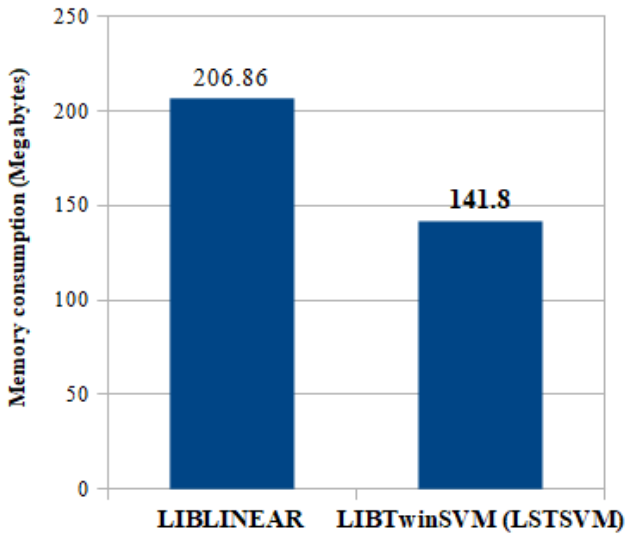
LightTwinSVM vs. LIBTwinSVM(TSVM)

About 4 times lower



LIBLINEAR vs. LIBTwinSVM(LSTSVM)

About 45 percent lower



Solved Problems

The efficiency of the optimizer component

- Created a C++ extension module.

Solved Problems

The efficiency of the optimizer component

- Created a C++ extension module.
- Used a caching technique.

Solved Problems

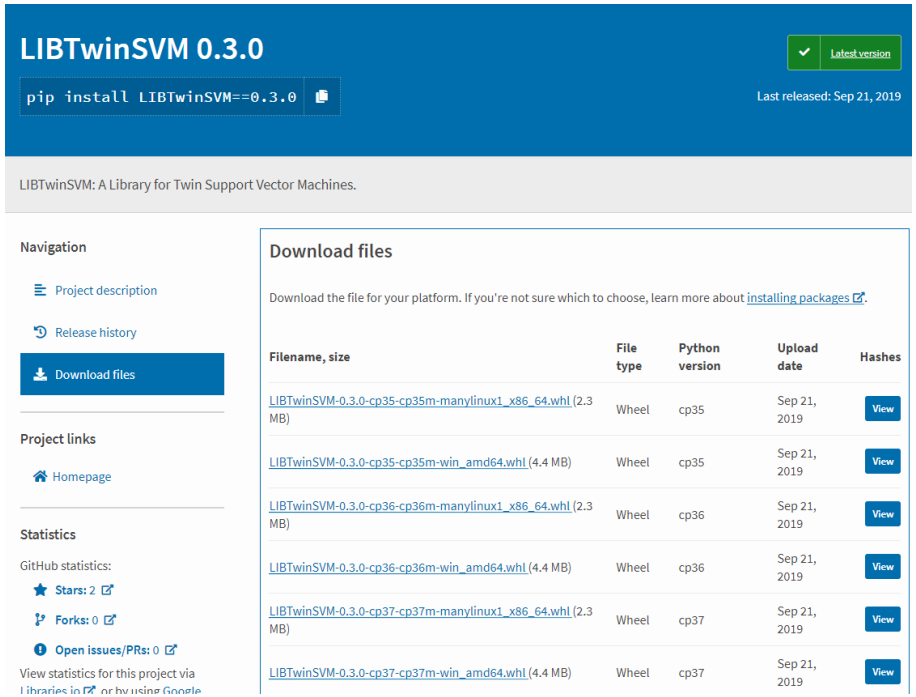
The efficiency of the optimizer component

- Created a C++ extension module.
- Used a caching technique.
- A NumPy array to an Armadillo matrix without memory copy.

Solved Problems

Pre-built Python wheels

- Including the dependencies of the C++ extension module.

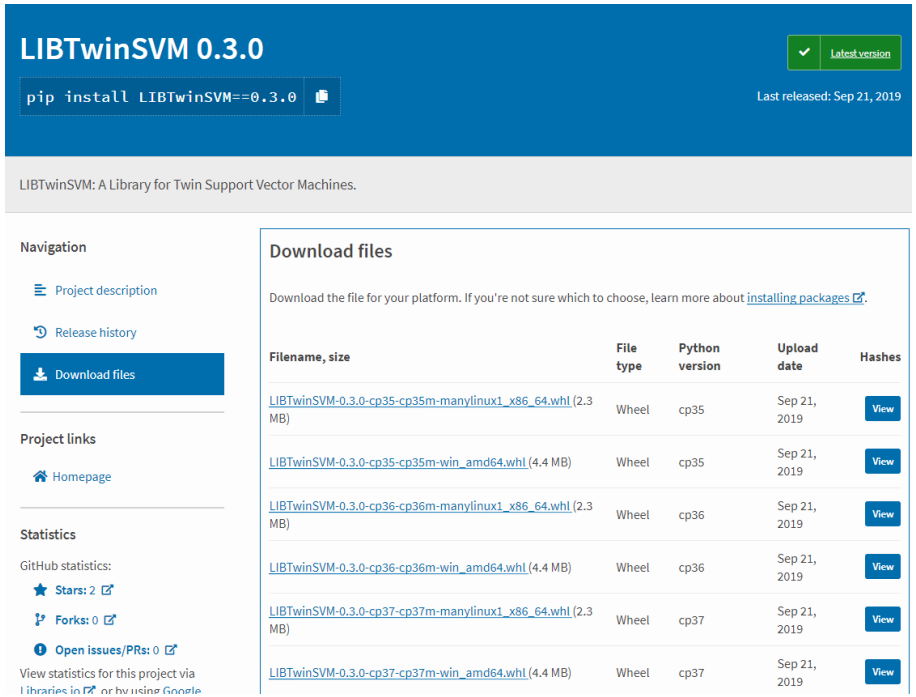


[10] https://github.com/matthew-brett/multibuild

Solved Problems

Pre-built Python wheels

- Including the dependencies of the C++ extension module.
- Used **multibuild**[10] project.

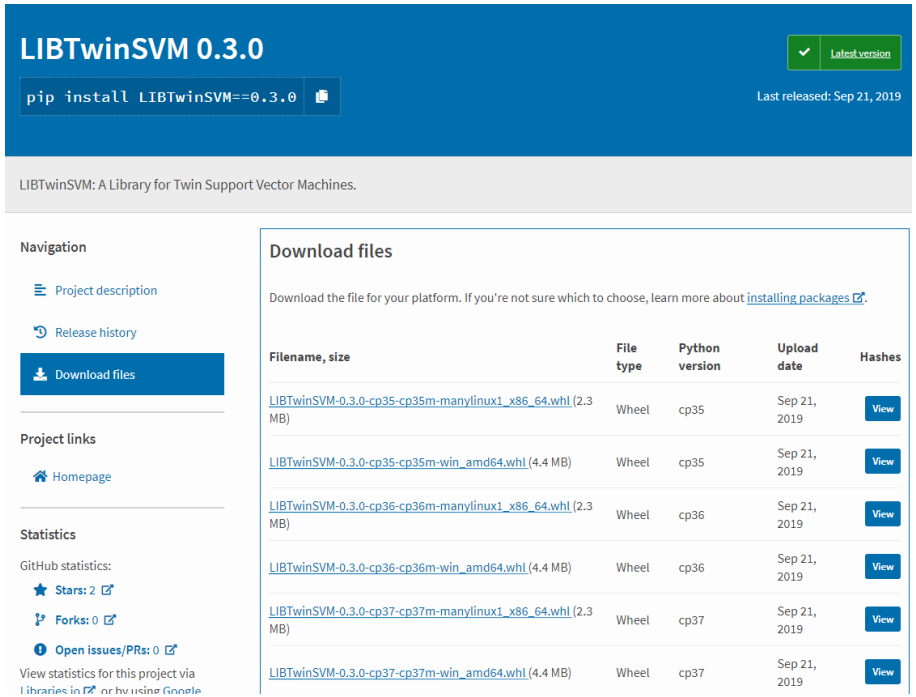


[10] https://github.com/matthew-brett/multibuild

Solved Problems

Pre-built Python wheels

- Including the dependencies of the C++ extension module.
- Used **multibuild**[10] project.
- Employed Travis CI and AppVeyor for generating the wheels.



[10] https://github.com/matthew-brett/multibuild

Solved problems

Very high memory consumption

Optimized the `fit()` of the TSVM classifier by changing the order of the computation.

Previous approach

1. Compute the matrix of the **first** optimization problem.
2. Compute the matrix of the **second** optimization problem.
3. Solve the first optimization problem.
4. Solve the second optimization problem.

Solved problems

Very high memory consumption

Optimized the `fit()` of the TSVM classifier by changing the order of the computation.

Previous approach

1. Compute the matrix of the **first** optimization problem.
2. Compute the matrix of the **second** optimization problem.
3. Solve the first optimization problem.
4. Solve the second optimization problem.

Current approach

1. Compute the matrix of the **first** optimization problem.
2. Solve the first optimization problem.
3. Free the matrix of the first optimization problem from the memory.
4. Compute the matrix of the **second** optimization problem.
5. Solve the second optimization

Open problems

- Support for GPUs.

Open problems

- Support for GPUs.
- The parallelization of the optimizer component.

Open problems

- Support for GPUs.
- The parallelization of the optimizer component.
- Support for sparse datasets.

Summary

- TSVM employs two non-parallel hyperplanes for classification.

Summary

- TSVM employs two non-parallel hyperplanes for classification.
- The main goal of the project: Simplicity and Ease of Use.

Summary

- TSVM employs two non-parallel hyperplanes for classification.
- The main goal of the project: Simplicity and Ease of Use.
- The library provides a GUI app and a Python API.

Summary

- TSVM employs two non-parallel hyperplanes for classification.
- The main goal of the project: Simplicity and Ease of Use.
- The library provides a GUI app and a Python API.
- The experimental results are comparable or even better.

Summary

- TSVM employs two non-parallel hyperplanes for classification.
- The main goal of the project: Simplicity and Ease of Use.
- The library provides a GUI app and a Python API.
- The experimental results are comparable or even better.
- There are open problems to improve the library.

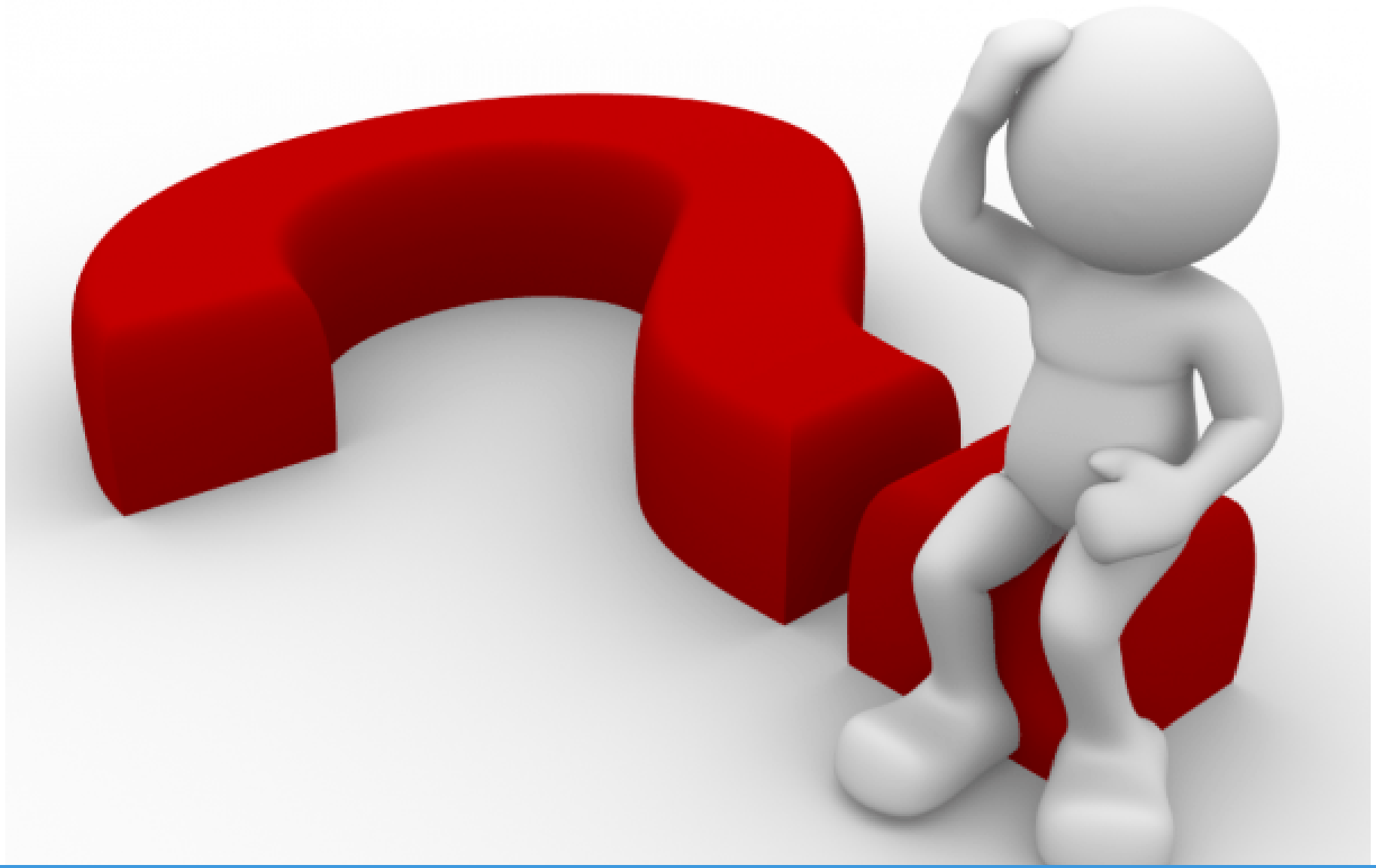
Quote

Quote

"We have to stop optimizing for programmers and start optimizing for users."

Jeff Atwood

Questions



References

Buitinck, L., G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, et al. (2013). "API design for machine learning software: experiences from the scikit-learn project". In: *arXiv preprint arXiv:1309.0238*.

Cortes, C. and V. Vapnik (1995). "Support-vector networks". In: *Machine learning* 20.3, pp. 273–297. ISSN: 0885-6125.

Khemchandani, R, S. Chandra, et al. (2007). "Twin support vector machines for pattern classification". In: *IEEE Transactions on pattern analysis and machine intelligence* 29.5, pp. 905–910.

Kumar, M. A. and M. Gopal (2009). "Least squares twin support vector machines for pattern classification". In: *Expert Systems with Applications* 36.4, pp. 7535–7543.

Mir, A. M. and J. A. Nasiri (2019). "LightTwinSVM: A Simple and Fast Implementation of Standard Twin Support Vector Machine Classifier". In: *Journal of Open Source Software* 4, p. 1252.

Musicant, D. (1998). "NDC: normally distributed clustered datasets". In: *Computer Sciences Department, University of Wisconsin, Madison*.

Peng, X., D. Chen, and L. Kong (2014). "A clipping dual coordinate descent algorithm for