

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
- Background
- Introduction
- Main features
- Design
- Implementation
- API
- Benchmarks
- Challenges and Problems
- Summary

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 Cortes¹ in 1995.
- Finds an optimal separating hyperplane.

¹C. 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 Cortes¹ in 1995.
- Finds an optimal separating hyperplane.

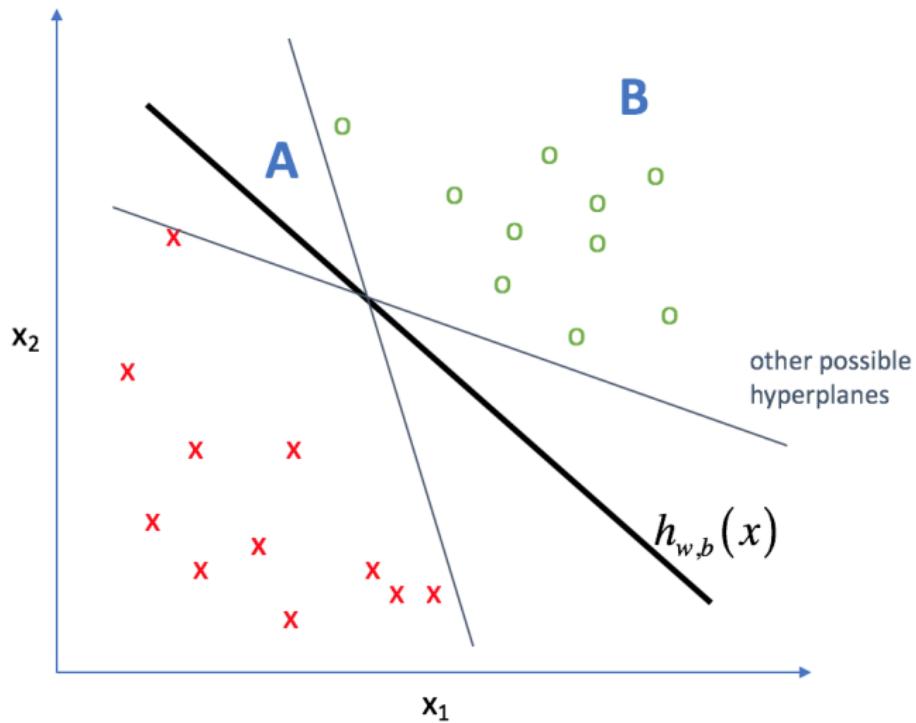
SVM's optimization problem

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

¹C. 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, TSVM² was proposed in 2007.
- Two non-parallel hyperplanes.

²R 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, TSVM² was proposed in 2007.
- Two non-parallel hyperplanes.

TSVM's optimization problems

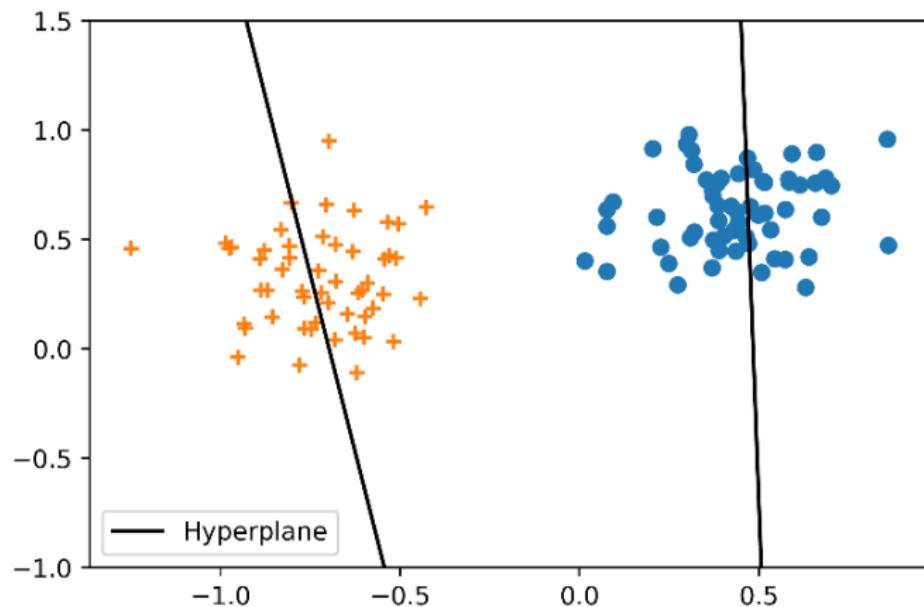
$$\begin{aligned} \min_{\mathbf{w}_1, b_1} \quad & \frac{1}{2} \|\mathbf{A}\mathbf{w}_1 + \mathbf{e}_1 b_1\|^2 + c_1 \mathbf{e}_2^T \xi \\ \text{s.t.} \quad & -(\mathbf{B}\mathbf{w}_1 + \mathbf{e}_2 b_1) + \xi \geq \mathbf{e}_2, \xi \geq 0 \end{aligned} \tag{2}$$

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

²R 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:**

³A. 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.

³A. 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.

³A. 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.

³A. 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.

³A. 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.

A Library for Twin Support Vector Machines

mir-am / LIBTwinSVM

Code Issues Pull requests Projects Wiki Security Insights Settings

262 commits 3 branches 3 releases 2 contributors GPL-3.0

Create new file Upload files Find file Choose or download

Branch: master New pull request

Latest commit 145944a on Sep 21

min-am Corrected the version number of the library.

benchmark Fixing memory problem of dplDCD optimizer - Part 2

dataset Added The characteristics of UCI benchmark datasets in docs. (skip ci)

docs Removed benchmarks in the docs temporarily. (skip ci)

libtsvm Corrected the version number of the library.

tests Skipping a test.

gfgmene Added clipDCD optimizer to the project.

gmmodules Added Amadillo library as submodule to the repo.

.travis.yml Running unit tests on Travis CI with pytest.

CHANGELOG.md Corrected the spell and grammar in the CHANGELOG file.

LICENSE.txt LICENSE.txt

MANIFEST.in Updated setup.py file with the long description of the project.

README.md A note for installing dependencies for Linux systems. (skip ci)

requirements.txt Modified Requirements.

setup.py Cleaned setup.py files.

Journal of Machine Learning Research (2019)

Submitted 5/31; Published xx/xx

LIBTwinSVM: A Library for Twin Support Vector Machines

Amir M. Mir^{†‡}

Mahdi Rahbar^{†‡}

Jalal A. Nasiri[†]

[†]*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-sm/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).

⁴M. 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

⁵X. 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.

⁴M. 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

⁵X. 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** (ClipDCD⁵) in C++.

⁴M. 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

⁵X. 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** (ClipDCD⁵) in C++.
- Supports **Linear**, Radial Basis Function (**RBF**), and **Rectangular** kernels.

⁴M. 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

⁵X. 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** (ClipDCD⁵) 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.

⁴M. 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

⁵X. 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** (ClipDCD⁵) 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.

⁴M. 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

⁵X. 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** (ClipDCD⁵) 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.

⁴M. 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

⁵X. 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

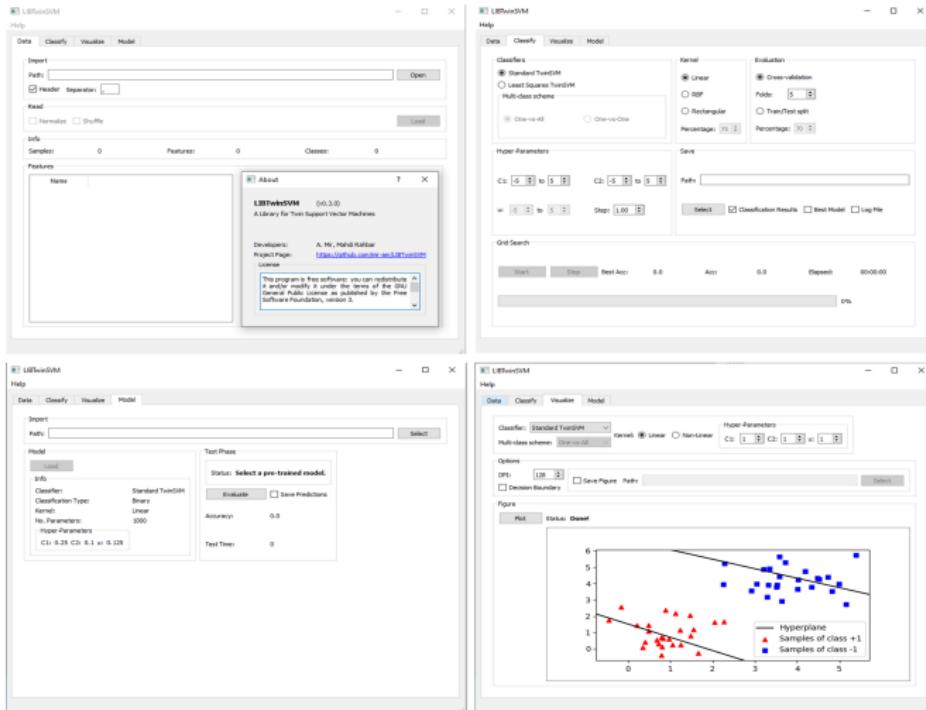
- A simple and user-friendly Graphical User Interface (GUI).
- Supports both **standard TSVM** and its **Least Squares extension**⁴.
- A **fast optimizer** (ClipDCD⁵) in C++.
- Supports **Linear**, Radial Basis Function (**RBF**), and **Rectangular** kernels.
- Supports popular **multi-class classification** 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 save the best-fitted model on the disk and also can load pre-trained models.

⁴M. 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

⁵X. 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

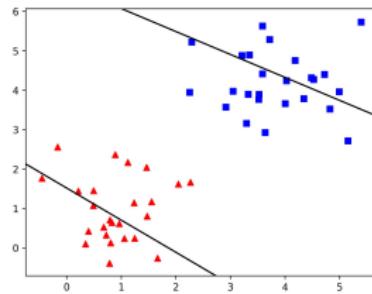
The GUI application



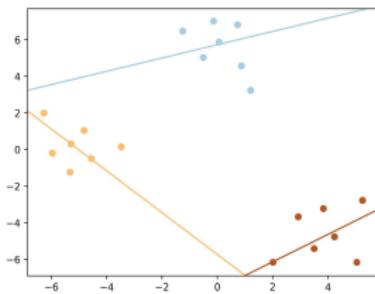
Main features of the LIBTwinSVM Library

The visualization feature

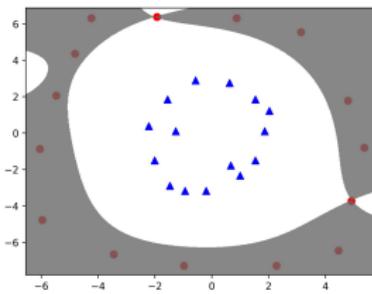
Binary TSVM with Linear Kernel



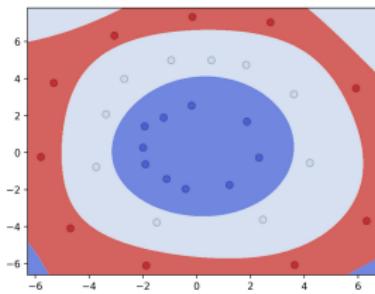
OVA-TSVM with Linear Kernel



Binary TSVM with Non-linear Kernel

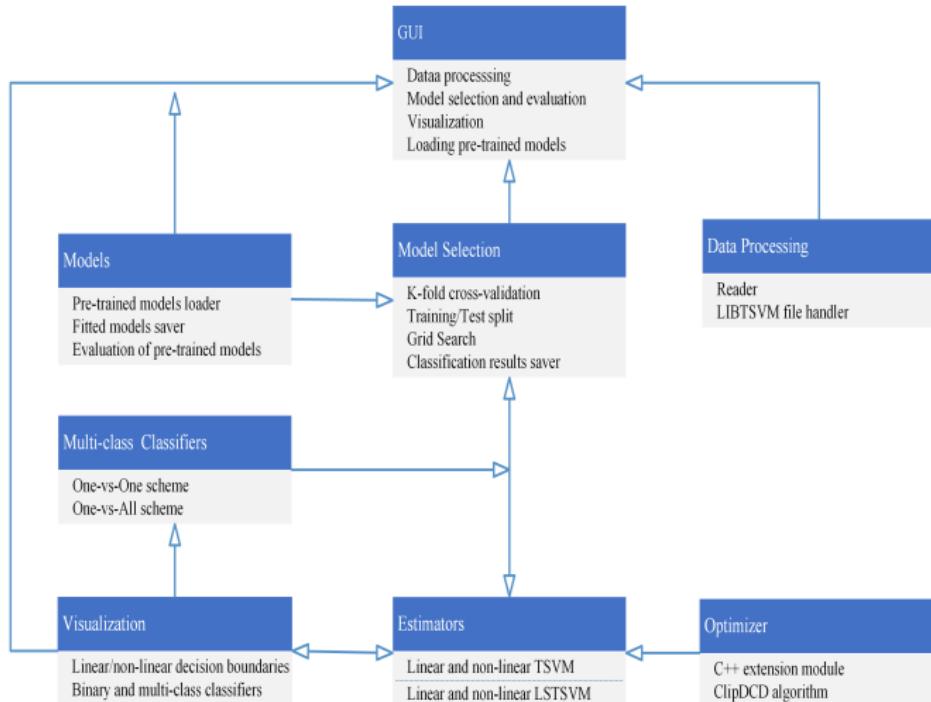


OVA-TSVM with Non-linear Kernel



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 CVXOPT⁶ package.

⁶<https://cvxopt.org/>

Implementation

Optimizer component

- It is the performance-critical part of the library.
- We could use CVXOPT⁶ 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 CVXOPT⁶ 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 CVXOPT⁶ 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 CVXOPT⁶ 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.

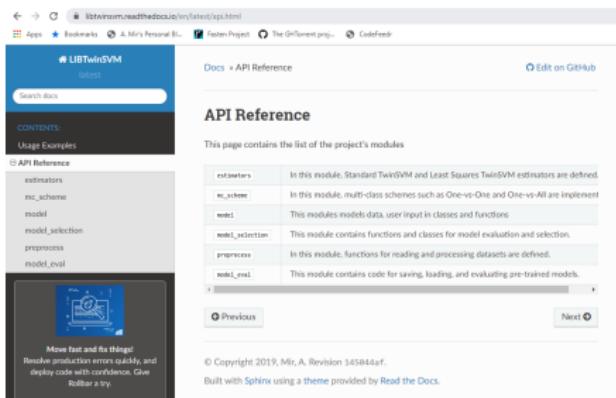
⁶<https://cvxopt.org/>

API

- Inspired by the design of Scikit-learn package⁷.

⁷L. 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*

- Inspired by the design of Scikit-learn package⁷.
- The interfaces of the estimators provides `fit()` and `predict()`.



⁷L. 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*

- Inspired by the design of Scikit-learn package⁷.
- The interfaces of the estimators provides `fit()` and `predict()`.
- API reference and usage examples are provided.

The screenshot shows a web browser displaying the API Reference for the LIBTwinSVM project. The URL in the address bar is `libtwinsvm.readthedocs.io/en/latest/api.html`. The page has a dark-themed header with the project name "LIBTwinSVM" and a "Latest" button. Below the header is a navigation bar with links for "CONTENTS", "Usage Examples", "API Reference", and "Edit on GitHub". The main content area is titled "API Reference" and contains a table of contents for the project's modules:

<code>estimators</code>	In this module, Standard TwinSVM and Least Squares TwinSVM estimators are defined.
<code>mc_scheme</code>	In this module, multi-class schemes such as One-vs-One and One-vs-All are implemented.
<code>model</code>	This module models data, user input in classes and functions.
<code>model_selection</code>	This module contains functions and classes for model evaluation and selection.
<code>preprocess</code>	In this module, functions for reading and processing datasets are defined.
<code>model_eval</code>	This module contains code for saving, loading, and evaluating pre-trained models.

At the bottom of the page, there are "Previous" and "Next" navigation links, and copyright information: "© Copyright 2019, Mir, A. Revision 145844af. Built with Sphinx using a theme provided by Read the Docs."

⁷L. 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*

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 = ',' # separator 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

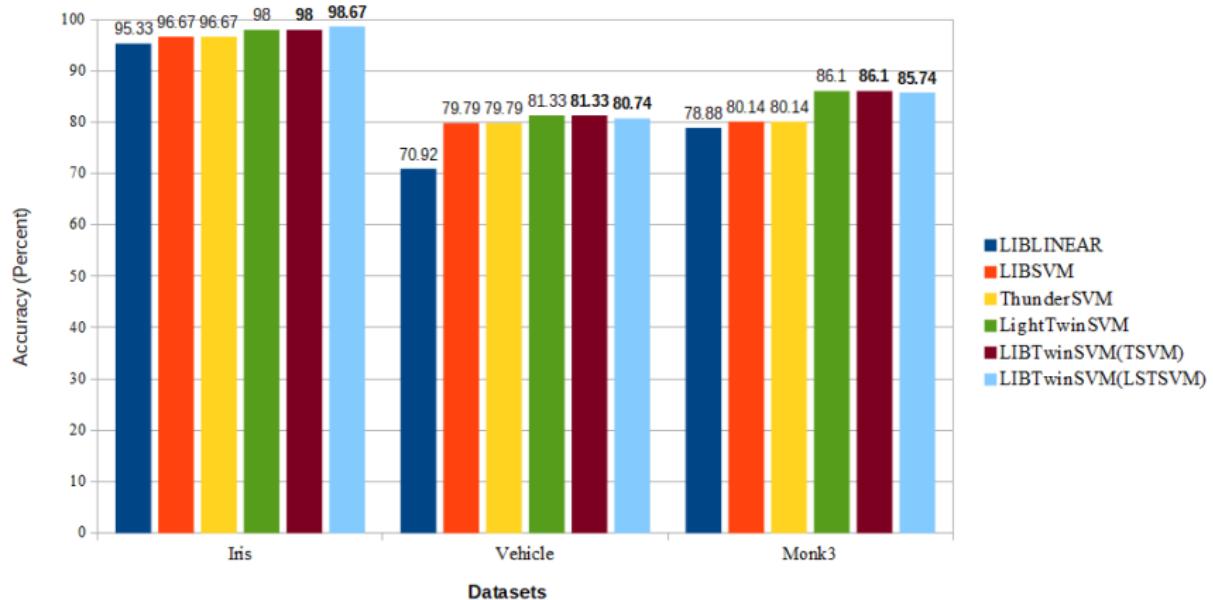
Experiment details:

- 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



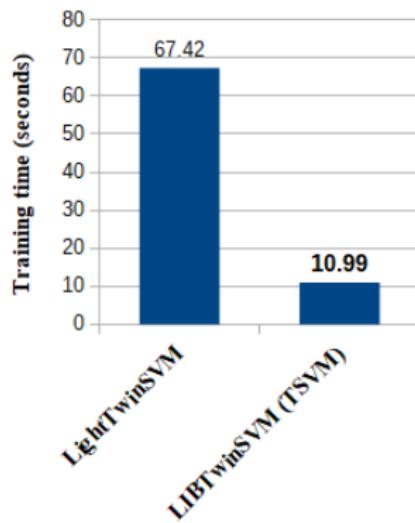
Benchmarks

Speed: Training time

NDC-50K dataset⁹

LightTwinSVM vs. LIBTwinSVM(TSVM)

About 6 times improvement



⁹ D. Marinelli (1999). "NDC: a new fully distributed clustered dataset". In: Computer Science Department, University of

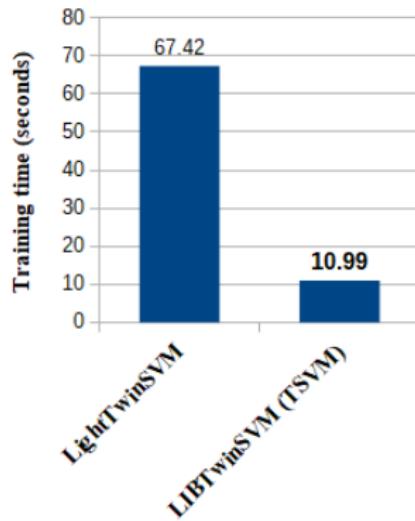
Benchmarks

Speed: Training time

NDC-50K dataset⁹

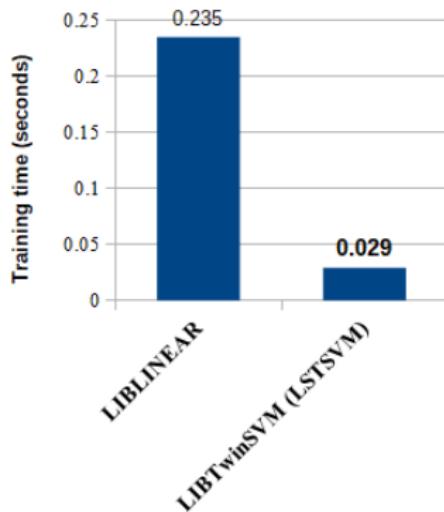
LightTwinSVM vs. LIBTwinSVM(TSVM)

About 6 times improvement



LIBLINEAR vs. LIBTwinSVM(LSTSVM)

About 8 times faster



⁹D. Marinelli (1999). "NDC: a new fully distributed clustered dataset". In: Computer Science Department, University of Siena.

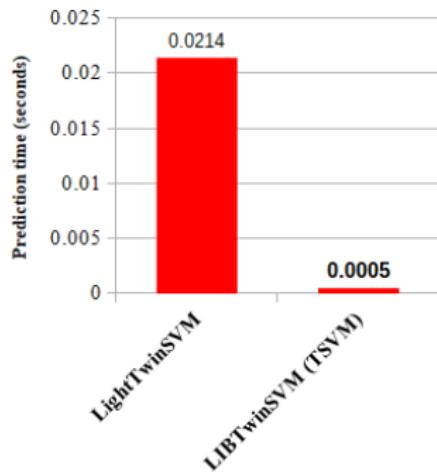
Benchmarks

Speed: Prediction time

NDC-50K dataset

LightTwinSVM vs. LIBTwinSVM(TSVM)

About 42 times improvement



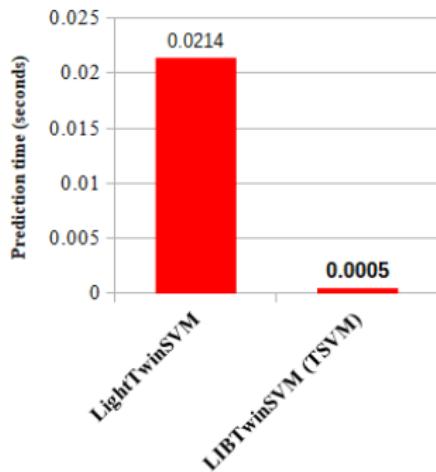
Benchmarks

Speed: Prediction time

NDC-50K dataset

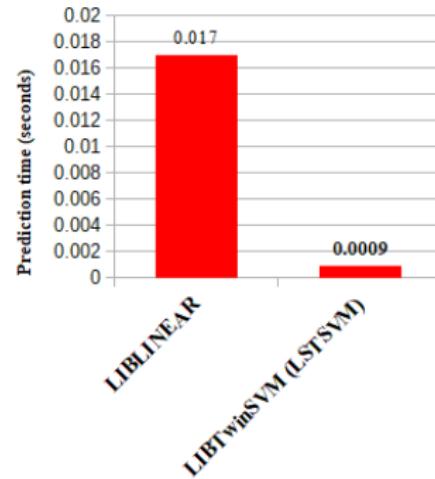
LightTwinSVM vs. LIBTwinSVM(TSVM)

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.

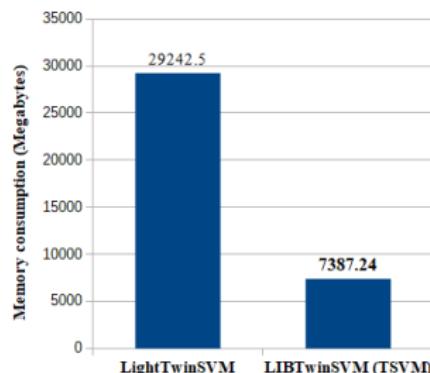
Benchmarks

Memory consumption

NDC-50K dataset

LightTwinSVM vs. LIBTwinSVM(TSVM)

About 4 times lower



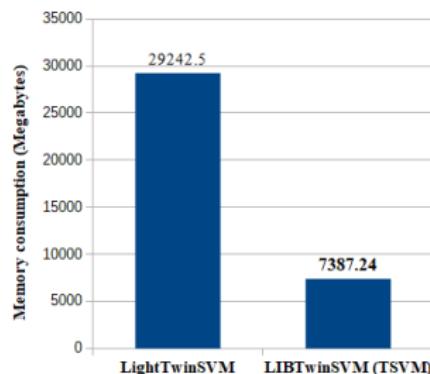
Benchmarks

Memory consumption

NDC-50K dataset

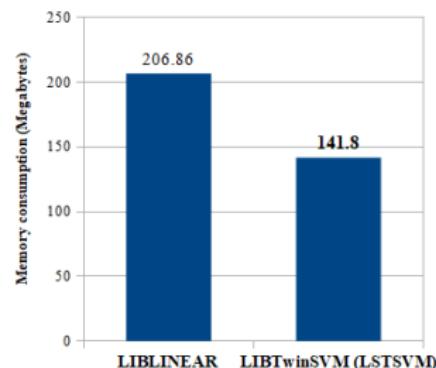
LightTwinSVM vs. LIBTwinSVM(TSVM)

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.

The screenshot shows the GitHub project page for LIBTwinSVM 0.3.0. At the top, there's a search bar with the query "LIBTwinSVM==0.3.0" and a "Search" button. To the right, it says "Last released: Sep 21, 2019". Below the search bar, there's a brief description: "LIBTwinSVM: A Library for Twin Support Vector Machines." On the left, there's a sidebar with "Navigation" links: "Project description", "Release history", and "Download files" (which is currently selected). Under "Project links", there's a "Homepage" link. In the "Statistics" section, it shows GitHub statistics: 2 stars, 0 forks, and 0 open issues. Below that, it says "View statistics for this project via i-headers in IP or via the Googles". The main content area is titled "Download files" and lists six pre-built Python wheels:

Filename, size	File type	Python version	Upload date	Hashes
LIBTwinSVM 0.3.0-cp35-cp35m-manylinux1_x86_64.whl (2.3 MB)	Wheel	cp35	Sep 21, 2019	View
LIBTwinSVM 0.3.0-cp35-cp35m-win_amd64.whl (4.4 MB)	Wheel	cp35	Sep 21, 2019	View
LIBTwinSVM 0.3.0-cp36-cp36m-manylinux1_x86_64.whl (2.3 MB)	Wheel	cp36	Sep 21, 2019	View
LIBTwinSVM 0.3.0-cp36-cp36m-win_amd64.whl (4.4 MB)	Wheel	cp36	Sep 21, 2019	View
LIBTwinSVM 0.3.0-cp37-cp37m-manylinux1_x86_64.whl (2.3 MB)	Wheel	cp37	Sep 21, 2019	View
LIBTwinSVM 0.3.0-cp37-cp37m-win_amd64.whl (4.4 MB)	Wheel	cp37	Sep 21, 2019	View

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.

The screenshot shows the GitHub project page for LIBTwinSVM 0.3.0. At the top, there's a search bar with the text "pip install LIBTwinSVM==0.3.0" and a "Search" button. To the right, it says "Last released: Sep 21, 2019". Below the search bar, the project title "LIBTwinSVM 0.3.0" is displayed in large blue text. A green button labeled "Latest version" is visible. The main content area has a light gray background. On the left, there's a sidebar with "Navigation" sections for "Project description", "Release history", and "Download files" (which is currently selected). Below that are "Project links" to "Homepage" and "Statistics". The "Statistics" section includes GitHub statistics: 2 stars, 0 forks, and 0 open issues. The main content area is titled "Download files" and contains a table with file details:

Filename, size	File type	Python version	Upload date	Hashes
LIBTwinSVM 0.3.0-cp35-cp35m-manylinux1_x86_64.whl (2.3 MB)	Wheel	cp35	Sep 21, 2019	View
LIBTwinSVM 0.3.0-cp35-cp35m-win_amd64.whl (4.4 MB)	Wheel	cp35	Sep 21, 2019	View
LIBTwinSVM 0.3.0-cp36-cp36m-manylinux1_x86_64.whl (2.3 MB)	Wheel	cp36	Sep 21, 2019	View
LIBTwinSVM 0.3.0-cp36-cp36m-win_amd64.whl (4.4 MB)	Wheel	cp36	Sep 21, 2019	View
LIBTwinSVM 0.3.0-cp37-cp37m-manylinux1_x86_64.whl (2.3 MB)	Wheel	cp37	Sep 21, 2019	View
LIBTwinSVM 0.3.0-cp37-cp37m-win_amd64.whl (4.4 MB)	Wheel	cp37	Sep 21, 2019	View

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.

The screenshot shows the GitHub project page for LIBTwinSVM 0.3.0. At the top, there's a search bar and a 'Login' button. Below that, a blue header bar contains the project name 'LIBTwinSVM 0.3.0' and a 'pip install LIBTwinSVM==0.3.0' button. To the right, it says 'Last released: Sep 21, 2019'. The main content area has a light gray background. On the left, there's a sidebar with 'Navigation' (Project description, Release history, Download files), 'Project links' (Homepage), and 'Statistics' (GitHub statistics: Stars: 2, Forks: 0, Open issues: PRs: 0). The main area is titled 'Download files' and contains a table with six rows of file information. Each row includes a 'View' button. The columns are: Filename, size, File type, Python version, Upload date, and Hashes.

Filename, size	File type	Python version	Upload date	Hashes
LIBTwinSVM 0.3.0-cp35-cp35m-manylinux1_x86_64.whl (2.3 MB)	Wheel	cp35	Sep 21, 2019	View
LIBTwinSVM 0.3.0-cp35-cp35m-win_amd64.whl (4.4 MB)	Wheel	cp35	Sep 21, 2019	View
LIBTwinSVM 0.3.0-cp36-cp36m-manylinux1_x86_64.whl (2.3 MB)	Wheel	cp36	Sep 21, 2019	View
LIBTwinSVM 0.3.0-cp36-cp36m-win_amd64.whl (4.4 MB)	Wheel	cp36	Sep 21, 2019	View
LIBTwinSVM 0.3.0-cp37-cp37m-manylinux1_x86_64.whl (2.3 MB)	Wheel	cp37	Sep 21, 2019	View
LIBTwinSVM 0.3.0-cp37-cp37m-win_amd64.whl (4.4 MB)	Wheel	cp37	Sep 21, 2019	View

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