Research Ideas and Outcomes 9: e101006
doi: 10.3897/rio.9.e101006

Software Description

pyRiemann-qiskit: A Sandbox for Quantum Classification Experiments with Riemannian Geometry

Anton Andreev‡, Grégoire H. Cattan§, Sylvain Chevallier|, Quentin Barthélemy¶

‡ GIPSA-lab, CNRS, University of Grenoble-Alpes, Grenoble, France
§ IBM Software, Krakow, Poland
| LISV, University of Paris-Saclay, Paris, France
¶ Foxteam, Vaulx-en-Velin, France

Corresponding author: Grégoire H. Cattan (gcattan@hotmail.fr)

Reviewed v 1

Academic editor: Editorial Secretary

Received: 25 Jan 2023 | Accepted: 15 Mar 2023 | Published: 20 Mar 2023

Citation: Andreev A, Cattan GH, Chevallier S, Barthélemy Q (2023) pyRiemann-qiskit: A Sandbox for Quantum Classification Experiments with Riemannian Geometry. Research Ideas and Outcomes 9: e101006. https://doi.org/10.3897/rio.9.e101006

Abstract

Background

Quantum computing is a promising technology for machine learning, in terms of computational costs and outcomes. In this work, we intend to provide a framework that facilitates the use of quantum machine learning in the domain of brain-computer interfaces – where biomedical signals, such as brain waves, are processed.

New information

To this end, we integrated *Qiskit*, a well-known quantum library, with *pyRiemann*, a framework for the analysis of biomedical signals using Riemannian Geometry. In this paper, we describe our approach, the main elements of our implementation and our research directions. A key result is the creation of a standardised pipeline (*QuantumClassifierWithDefaultRiemannianPipeline*) for the binary classification of brain

waves. The git repository reported in this paper also contains a complete test suite and examples to guide practitioners. We believe that this software will enable further research on the joint field of brain-computer interfaces and quantum computing.

Keywords

information geometry, machine learning, time series, signal processing, Brain-Computer Interface (BCI)

Introduction

Quantum computing has its roots in the so-called double-slit experiment conducted by Thomas Young in 1802. In this experiment, a small entity, such as a photon or an electron, is directed towards two parallel slits and the resulting interference pattern is observed. The observation shows that the entity behaves as a wave, which suggests that it passes through both slits simultaneously. From a computational perspective, this wave-particle duality means that a single bit of information can be encoded as a quantum bit, which is a superposition of two different states. This unique characteristic of quantum computing offers a significant advantage over classical computing in terms of computational time and outcomes, such as for pattern recognition or when using limited training sets (Rebentrost et al. 2014, Blance and Spannowsky 2021).

A ubiquitous library for quantum computing is *Qiskit* (Abraham 2019). Qiskit is authored by IBM and distributed under the Apache 2.0 licence which provides both quantum algorithms and backends. A backend can be either a local machine or a remote machine, which can be an emulated one or a real quantum computer. Qiskit abstraction over the type of the selected machine makes designing and using quantum algorithms seamless.

Qiskit implements quantum versions of two support vector-like classifiers, named quantum-enhanced support vector classifier (QSVC) and variational quantum classifier (VQC, Havlíček et al. (2019)). In practice, experiments on artificial datasets suggest that quantum-enhanced support vector machines (SVMs) offer a provable speed-up compared to classical algorithms (Liu et al. 2021). These classifiers likely offer an advantage over classical SVMs in situations where the classification task is complex. Task complexity depends on the available data, the quality of the data and the encoding of the data into quantum states.

pyRiemann-qiskit implements a wrapper around QSVC and VQC, enabling quantum classification, based on Riemannian Geometry (RG).

Project description

Funding: This research received no specific grant from any funding agency in the public, commercial or not-for-profit sectors.

Web location (URIs)

Homepage: https://doi.org/10.5281/zenodo.7565763

Wiki: https://github.com/pyRiemann/pyRiemann-qiskit/wiki

Download page: https://pyriemann-qiskit.readthedocs.io/en/latest/installing.html

Bug database: https://github.com/pyRiemann/pyRiemann-qiskit/issues

Repository

Type: Git

Browse URI: https://github.com/pyRiemann/pyRiemann-qiskit/

Usage licence

Usage licence: Other

IP rights notes: BSD 3-Clause "New" or "Revised" License

Implementation

Implements specification

pyRiemann-qiskit facilitates the creation and parameterisation of a quantum backend and is fully compliant with scikit-learn's transformers, estimators and classifiers, so it becomes easy to integrate quantum classification into existing pipelines. It also supports docplex for the specification of convex optimisation problems, with the limitation of using binary and unconstrained variables. pyRiemann-qiskit is built on top of *pyRiemann* (Barachant et al. 2020), a machine learning library based on RG, thereby enabling the manipulation of covariance matrices (and, to a larger extent, semi-positive definite matrices) within the Riemannian manifold and which allows the projection of these matrices into a tangent space at a specific point on the Riemannian manifold. Classification is performed either in the tangent space or by directly utilising the Riemannian distance between each sample (represented by its covariance matrix) and a class prototype in the manifold.

pyRiemann-qiskit also includes examples to guide practitioners, as well as a complete test suite. We will briefly describe below the functionalities provided by the software.

Support for quantum classifiers

The software supports QSVC and VQC classifiers. The first concern regarding quantum classifiers is the encoding of classical data into quantum states. This operation is known as feature mapping. To obtain an advantage over classical computing, feature mapping must

implement quantum circuits which are difficult to emulate on a classical computer. Feature mapping is common in VQCs and QSVCs. Both are SVM-like classifiers in the sense that they generate a separating hyperplane. The difference between them is that VQCs uses a variational quantum circuit (also known as a variational form) for this task, whereas QSVCs uses a quantum-enhanced kernel with a conventional SVM. The software also supports the Pegasos implementation of QSVC, which offers a speed-up compared to QSVC (Gentinetta et al. 2022).

Table 1 demonstrates how to instantiate the VQC or QSVC classifier in pyRiemann-Qiskit.

Table 1.

Instantiation of VQC and QSVC classifiers in pyRiemann-qiskit.

```
from pyriemann_qiskit.classification import QuanticSVM, QuanticVQC
vqc = QuanticVQC()
qsvc = QuanticSVM()
pegasos = QuanticSVM(pegasos=True)
svc = QuanticSVM(quantum=False)
```

By default, the backend will be a local quantum simulator. However, it is possible to register on IBM quantum and request a free token to use one of the publicly available quantum computers. All classifiers accept a *q_account_token* parameter which, if valid, provides access to one of several real quantum computers available with IBM.

However, note that, at the time of writing, the number of qubits (and, therefore, the feature dimension) is limited to:

- 36 on a local quantum simulator;
- 5000 on a remote quantum simulator;
- 5–7 on a real free quantum computer;
- 127 on exploratory quantum computers (not available for public use).

Support for convex optimisation problems

The MDM algorithm (Barachant et al. 2012) consists in finding the minimum distance between a trial and a class prototype before labelling the trial with the prototype which is the closest to the trial. Alternatively, this task can be defined as a convex optimisation problem and executed as a quantum circuit.

pyRiemann-qiskit supports the docplex library for the definition of convex optimisation problems. Specifically, this is useful in these two cases:

- in training: computing the barycenter of the covariance matrices, i.e. a matrix is used as a point in the manifold and thus an average – called a class prototype – of all these points can be calculated;
- in classification: finding the minimum distance between a trial and a class prototype can also be defined as a quadratic optimisation problem (Zhao et al. 2019) – see section Direct classification of covariance matrices under Additional Information.

To calculate the class prototype, we need to select a *distance* and provide an optimiser. We model the distance as a convex problem, based on frobenius distance (python method *fro_mean_convex*). The model objective is to find the matrix which is at minimum distance to all covariance matrices. This matrix is then designated as the barycenter or class prototype of the covariance matrices. The optimiser is the component which is responsible for finding the coefficients of this barycenter, based on the model objective and the covariance matrices used as inputs. For the optimiser, pyRiemann-qiskit relies on Qiskit implementation of QAOA (Quantum Approximate Optimisation Algorithm). However, QAOA is limited to solving only QUBO (Quadratic Unconstrained Binary Optimisation) problems, that is, problems with unconstrained and binary variables only. In other words, the mean matrix, being the subject of optimisation, can only accept binary variables and, thus, the input covariance matrices can only contain zeros and ones as coefficients. For this reason, pyRiemann-qiskit uses a wrapper around Qiskit's QAOA optimiser (class NaiveQAOAOptimizer) that rounds covariance matrices to a certain precision and transforms each resulting integer coefficient to binary coefficients. This transformation is based on Qiskit's *IntegerToBinary*, a bounded coefficient encoding method.

The complexity of the QAOA optimiser rises as a function of the size of the covariance matrices and the upper-bound coefficient. The size of the covariance matrices depends on the number of input channels in the input time epoch, as well as the dimension reduction method that is used. The upper-bound coefficient also has an impact on the final size of the covariance matrices. The upper-bound coefficient is simply the maximal value that can possibly take a coefficient in a matrix. If all coefficients inside a matrix are integers ranging from 0 to 3, then each of them can be represented by only 2 bits (00, 01, 10, 11). In this case, each integer coefficient inside the matrix can be replaced by two binary coefficients and, thus, the size of the resulting matrices will be "only" twice as large. As the mean matrix to optimise has the exact same size as the input covariance matrices, it means that there will be two times more variables to optimise inside the mean matrix. Therefore, this implies a higher number of qubits to hold the variables inside the matrix. As QAOA is computationally expensive, we also provide a wrapper over the classical optimiser Cobyla that allows for easier testing.

Table 2 demonstrates how to use the fro_mean_convex method.

Table 2.

Convex distance optimisation with frobenius mean in pyRiemann-qiskit.

```
from pyriemann.utils.distance import distance_methods
from pyriemann.classification import MDM
from pyriemann.estimation import XdawnCovariances
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import StratifiedKFold, cross_val_score
from pyriemann_qiskit.utils.mean import fro_mean_convex
metric = {'mean': "convex",'distance': "convex"}
distance_methods["convex"] = lambda A, B: np.linalg.norm(A - B, ord='fro')
clf = make_pipeline(XdawnCovariances(), MDM(metric=metric))
```

```
skf = StratifiedKFold(n_splits=5)
n_matrices, n_channels, n_classes = 100, 3, 2
covset = get_covmats(n_matrices, n_channels)
labels = get_labels(n_matrices, n_classes)
score = cross_val_score(clf, covset, labels, cv=skf, scoring='roc_auc')
```

If the MDM classifier is supplied with the "convex" metric, it will automatically use the fro_mean_convex method for computing the mean – the class prototype. The default optimiser for the fro_mean_convex method is the Cobyla optimiser.

Classification of vectorised covariances matrices

The number of qubits is limited (especially on real hardware) and this puts a limit on the length of the feature vectors that can be used with quantum programming. In *pyRiemannQiskit*, we use the method of *tangent space* vectorisation. It consists of creating covariance matrices from the signal then projecting them into the tangent space of the Riemannian manifold. The result is a feature vector with reduced dimensionality. The dimensionality can be further reduced using PCA, for example, in order to match the number of available qubits.

The code snippet below (Table 3) demonstrates how to perform a dimension reduction of the signal epoch using Xdawn (Rivet and Souloumiac 2013), apply the tangent space method, then further reduce the size of the feature space to match the capability of the quantum backend in charge of running our quantum classifier (here *QuanticSVM*).

Table 3.

Example of quantum classification pipeline with pyRiemann-qiskit.

```
from pyriemann.estimation import XdawnCovariances
from pyriemann.classification import TangentSpace
from sklearn.decomposition import PCA
from pyriemann_qiskit.classification import QuanticSVM
pipe = make_pipeline(XdawnCovariances(nfilter=2), TangentSpace(), PCA(), QuanticSVM())
```

For convenience, the library provides the *QuantumClassifierWithDefault RiemannianPipeline* class, which implements the above pipeline.

Audience

pyRiemann-qiskit is a sandbox for experimenting with quantum computing in conjunction with RG. It unifies within the same library a quantum computational library and RG tools to ultimately allow for the creation of better Brain-Computer Interfaces (BCI). RG tools are proved to perform very well on the classification of EEG ERP signals (Lotte et al. 2018). Therefore, the primary audience we target is practitioners coming from the BCI field, willing to experiment with quantum computing for the classification of electroencephalography (EEG) or magnetoencephalography signals. An initial study on this topic is available in Cattan and Andreev (2022). It can be downloaded from pyRiemann-qiskit. However, note

that the tools provided by the library are also relevant to other domains, such as classification of other biometric signals, image recognition or detection of fraud transactions (e.g. Grossi et al. (2022)).

In summary, we seek to encourage the use of quantum computing together with RG for concrete (BCI) applications, opening new and interesting research paths. For example, it would be interesting to investigate BCI illiteracy, a situation in which classical classifiers usually fail to generalise well using EEG data.

Additional information

Future work directions on the software include the direct classification of covariance matrices and multi-class classification.

Direct classification of covariance matrices

The MDM algorithm consists of finding the minimum distance between a trial and a class prototype before labelling the trial with the class that is the closest to the trial. It is a decision optimisation problem that can be solved using Qiskit's QAOA, under the conditions that: 1) it is provided in the form of a convex model and, 2) it is quadratic, unconstrained and contains only binary variables.

For instance, MDM, based on the Log-Euclidian metric, has the following expression (Zhao et al. 2019):

$$\arg\min w^T D w - 2 Vec(\log Y) D$$

with $\sum w_i = 1, w_i >= 0 \forall i$ and $D = [Vec(\log X_1)...Vec(\log X_N)]$. *D* is the training data, *w* are the weights that are being optimised, *Y* is the new trial to classify and X_i is a class prototype. Class prototypes are built during training using the mean covariance matrices and, therefore, this approach is compatible with the fro_mean_convex method previously introduced.

Note that the equation above is a quadratic optimisation problem. However, weights in the *w* vector are constrained continuous variables, thus complicating the use of the IntegerToBinary approach.

Besides, the equation must be solved for each new trial that needs to be classified. The complexity of determining the correct weights to minimise the equation varies as a function of the number of classes and the upper bound coefficient which is used for the IntegerToBinary method (the higher the coefficient, the higher the complexity). While potentially slower, this quantum-optimised version of the MDM algorithm *could* produce better results, especially in cases where classical computation fails.

Multi-class classification

At the time of writing, pyRiemann-qiskit only supports binary classification of covariance matrices (i.e. presence or absence of an ERP). Further work envisions the implementation of multi-class classifiers.

References

- Abraham H, et al. (2019) Qiskit: An Open-source Framework for Quantum Computing. Zenodo. 10.5281/zenodo.2562110. https://doi.org/10.5281/zenodo.2562110
- Barachant A, Bonnet S, Congedo M, Jutten C (2012) Multiclass brain-computer interface classification by Riemannian geometry. IEEE transactions on bio-medical engineering 59 (4): 920-928. https://doi.org/10.1109/TBME.2011.2172210
- Barachant A, Barthélemy Q, Gramfort A, King J-, Chevallier S, P. L. Rodrigues, Berg GWv (2020) pyRiemann. Zenodo. URL: https://doi.org/10.5281/zenodo.593816
- Blance A, Spannowsky M (2021) Quantum machine learning for particle physics using a variational quantum classifier. Journal of High Energy Physics 2021 (2). https://doi.org/10.1007/JHEP02(2021)212
- Cattan G, Andreev A (2022) First steps to the classification of ERPs using quantum computation. NTB Berlin 2022 - International Forum on Neural Engineering & Brain Technologies. URL: https://hal.archives-ouvertes.fr/hal-03672246
- Gentinetta G, Thomsen A, Sutter D, Woerner S (2022) The complexity of quantum support vector machines. arXiv. Comment: 24 pages, 13 figures. URL: http://arxiv.org/abs/2203.00031
- Grossi M, Ibrahim N, Radescu V, Loredo R, Voigt K, Von Altrock C, Rudnik A (2022) Mixed Quantum-Classical Method For Fraud Detection with Quantum Feature Selection. arXiv. Comment: 11 pages, 12 figures, 9 tables. https://doi.org/10.48550/arXiv.2208.07963
- Havlíček V, Córcoles A, Temme K, Harrow A, Kandala A, Chow J, Gambetta J (2019) Supervised learning with quantum-enhanced feature spaces. Nature 567 (7747): 209-212. https://doi.org/10.1038/s41586-019-0980-2
- Liu Y, Arunachalam S, Temme K (2021) A rigorous and robust quantum speed-up in supervised machine learning. Nature Physics 17 (9): 1013-1017. https://doi.org/10.1038/s41567-021-01287-z
- Lotte F, Bougrain L, Cichocki A, Clerc M, Congedo M, Rakotomamonjy A, Yger F (2018) A Review of Classification Algorithms for EEG-Based Brain-Computer Interfaces: A 10 Year Update. Journal of Neural Engineering 15 (3). https://doi.org/10.1088/1741-2552/aab2f2.
- Rebentrost P, Mohseni M, Lloyd S (2014) Quantum Support Vector Machine for Big Data Classification. American Physical Society URL: https://dspace.mit.edu/handle/1721.1/90391
- Rivet B, Souloumiac A (2013) Optimal linear spatial filters for event-related potentials based on a spatio-temporal model: Asymptotical performance analysis. Signal Processing 93 (2): 387-398. https://doi.org/10.1016/j.sigpro.2012.07.019
- Zhao K, Wiliem A, Chen S, Lovell B (2019) Convex Class Model on Symmetric Positive Definite Manifolds. arXiv:1806.05343 [cs] https://doi.org/10.48550/arXiv.1806.05343