Performance Analysis of Support Vector Machine Implementations on the D-Wave Quantum Annealer

Harshil Singh Bhatia[1] and Frank Phillipson[2]

[1] Department of Computer Science and Engineering, Indian Institute of Technology, Jodhpur, India
[2] TNO, the Netherlands Organisation for Applied Scientific Research, The Hague, The Netherlands

Abstract. In this paper a classical classification model, Kernel-Support Vector machine, is implemented as a Quadratic Unconstrained Binary Optimisation problem. Here, data points are classified by a separating hyperplane while maximizing the function margin. The problem is solved for a public Banknote Authentication dataset and the well-known Iris Dataset using a classical approach, simulated annealing, direct embedding on the Quantum Processing Unit and a hybrid solver. The hybrid solver and Simulated Annealing algorithm outperform the classical implementation on various occasions but show high sensitivity to a small variation in training data.

Keywords: Quadratic Unconstrained Binary Optimisation, Quantum Annealing, Support Vector Machine, Performance Analysis

1 Introduction

With the advancement in quantum computing and machine learning in recent times, the combination of both fields has emerged, termed *Quantum Machine Learning* [20]. Quantum computing is a technique of using quantum mechanical phenomena, such as superposition and entanglement, for solving computation problems. The current technology has established two paradigms: gate-based quantum computers and quantum annealers. The size of these computers is still limited, with the state of the art gated model-based quantum computer having 72 qubits (Google's Bristlecone) and a quantum annealer having 5000 qubits (D-Wave's Advantage [10]). Also various other firms, such as IBM, Rigetti, Qutech and IonQ, are working towards a practically usable quantum computer. To this end, we need to work on quantum algorithms and quantum software engineering skills [24].

Support Vector Machines (SVMs) [26] are supervised learning models that are used for classification and regression analysis. SVMs are known for their stability, i.e., they don't produce largely different classifications with a small

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_7

variation in training data. SVMs are preferred over Deep Learning algorithms when a relatively small training data set is available.

The SVM can be implemented on a gated model-based quantum computer with time complexity logarithmic in the size of vectors and training samples. This idea was proposed by Rebentrost et al. [25]. In [18], a quantum support vector machine was experimentally realised on a four-qubit NMR (nuclear magnetic resonance) test bench. The work of [17] establishes quantum annealing as an effective method for classification of certain simplified computational biology problems. The quantum annealer showed a slight advantage in classification performance and nearly equalled the ranking performance of the other state of the art implementations, for fairly small datasets.

In [27], Willsch et al. use a non-linear kernel to train a model on the D-Wave 2000 Quantum Annealer. For small test cases, the quantum annealer gave an ensemble of solutions that often generalise better for the classification of unseen data, than a single global minimum provided, which is given by a classically implemented SVM. In [23] this approach is used, together with two other machine learning approaches, to classify mobile indoor/outdoor locations. One of their findings is that the quantum annealing results show a solution with more confidence than the simulated annealing solution.

One of the significant limitations of classical algorithms using a non-linear kernel is that the kernel function has to be evaluated for all pairs of input feature vectors which can be of high dimensions. In [4], Chatterjee et al. propose using generalised coherent states as a calculation tool for quantum SVM and the realization of generalised coherent states via experiments in quantum optics indicate the near term feasibility of this approach.

In this paper, we use the formulation proposed by Willsch et al. [27] for the SVM modeled as a Quadratic Unconstrained Binary Optimisation (QUBO) problem, and look at its implementation and performance on the 5000 qubits [10] quantum annealer manufactured by D-Wave Systems, both directly on the chip and using the hybrid solver offered by D-Wave Systems. The aim of the paper is to study the difference in behaviour of the quantum and classical algorithm for fixed hyperparamaters. The code for the implementation using DWave Samplers is available at [1].

This paper is structured as follows. In Section 2, we give a short introduction on quantum annealers. The problem description and its formulation are given in Section 3. Section 4 deals with the various implementations of the Support Vector Machine. The classification results are presented in Section 5. We end with some conclusions and future research prospects.

2 Annealing Based Quantum Computing

Quantum annealing is based on the work of of Kadowaki and Nishimori [15]. The idea of quantum annealing is to create an equal superposition over all possible states. Then, by slowly turning on a problem-specific magnetic field, the qubits interact with each other and move towards the state with the lowest energy.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_7

The challenging task of quantum annealing is to formulate the desired problem in such terms that it corresponds to finding a global minimum, such that it can also be implemented on the hardware of the quantum device. The most advanced implementation of this paradigm is the D-Wave quantum annealer. This machine accepts a problem formulated as an Ising Hamiltonian, or rewritten as its binary equivalent, in QUBO formulation. The QUBO, Quadratic Unconstrained Binary Optimisation problem [12], is expressed by the optimisation problem:

$$\text{QUBO: } \min_{x \in \{0,1\}^n} x^t Q x, \tag{1}$$

where $x \in \{0,1\}^n$ are the decision variables and Q is a $n \times n$ coefficient matrix. QUBO problems belong to the class of NP-hard problems [19]. Many constrained integer programming problems can easily be transformed to a QUBO representation. For a large number of combinatorial optimisation problems the QUBO representation is known [12,19].

Next, this formulation needs to be embedded on the hardware. In the most advanced D-Wave Advantage version of the system, the 5000 qubits are placed in a Pegasus architecture [10] containing $\mathcal{K}_4$ and $\mathcal{K}_{6,6}$ sub-graphs. Pegasus qubits have a degree 15, i.e., they are externally coupled to 15 different qubits. The QUBO problem has to be transformed to this structure. Due to the current limitation of the chip size, a compact formulation of the QUBO and an efficient mapping to the graph is required. This problem is known as Minor Embedding. Minor Embedding is an NP-Hard problem and is automatically handled by the D-Wave's system [5]. Sometimes, fully embedding a problem on the Quantum Processing Unit (QPU) is difficult or simply not possible. In such cases, the D-Wave system employs built-in routines to decompose the problem into smaller sub-problems that are sent to the QPU, and in the end reconstructs the complete solution vector from all sub-sample solutions. The first decomposition algorithm introduced by D-Wave was *qbsolv* [2], which gave a possibility to solve problems of a large scale on the QPU. Although *qbsolv* was the main decomposition approach on the D-Wave system, it did not enable customisations of the workflow, and therefore is not particularly suited for all kinds of problems. The new decomposition approaches that D-Wave offers are D-Wave Hybrid [10] and the Hybrid Solver Service [8], offering more customization options of the workflow.

3 Problem description

Support Vector Machines are supervised learning models that are used for classification and regression analysis. The training set is given as $\{x_1, x_2, \ldots x_n\}$ that are d-dimensional vectors in a some space $\chi \in \mathcal{R}^d$, where d is the dimension of each vector, i.e., the number of attributes of the training data. We are also given their labels $\{y_1, y_2, ..., y_n\}$, where $y_i \in \{1, -1\}$. SVMs use hyperplanes that separate the training data by a maximal margin. In general, SVMs allow one to project the training data in the space χ to a higher dimensional feature space

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_7

$\mathcal{F}$ by $z = \Phi(x)$, where $z \in \mathcal{F}$. To find the optimal separating hyperplane having the maximum margin, the algorithm minimizes the following equation:

$$\min \frac{1}{2}||w||^2, \tag{2}$$

$$\text{subject to:} \quad y_i(w^T\Phi(x_i) + b) \geq 1, \quad \forall i = 1, ..., n, \tag{3}$$

where w is the normal vector for the separating hyperplane given by the equation, $w = \sum_i \alpha_i y_i \mathcal{K}(x_i, x)$, which can be transferred into its dual form by maximising its primal Lagrangian. This is further formulated as a quadratic programming problem:

$$\text{minimise} \left(\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathcal{K}(x_i, x_j) - \sum_{i=1}^{n} \alpha_i \right), \tag{4}$$

$$\text{subject to } 0 \leq \alpha_i \leq C \quad \forall i = 1, ..., n, \tag{5}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \tag{6}$$

where α_i is the weight assigned to the training sample x_i. If $\alpha_i > 0$, then x_i is a support vector. C is a regularisation parameter that controls the trade-off between achieving a low training error and a low testing error such that a generalization can be obtained for unseen data. The function $\mathcal{K}(x_i, x_j) = \Phi(x_i)^T\Phi(x_j)$ is Mercer's kernel function which allows us to calculate the dot product in high-dimensional space without explicitly knowing the non-linear Mapping. There are different forms of kernel functions, however, the SVM with a Gaussian Kernel (or RBF-kernel) has been popular because of its ability to handle cases with non-linear relation between classes and features and doing so while having less parameters. The Gaussian Kernel is defined as

$$\mathcal{K}(x_i, x_j) = e^{-\gamma||x_i - x_j||^2} \tag{7}$$

where $\gamma > 0$ is the hyperparameter.

The coefficients define a decision boundary that separates the vector space in two regions, corresponding to the predicted class labels. The decision function is fully specified by the support vectors and is used to predict samples around the optimal hyperplane. It is formulated as follows:

$$f(x) = w^T\Phi(x) + b = \sum_{i=1}^{n} \Phi(x_i)\Phi(x) + b, \tag{8}$$

$$f(x) = \sum_{i=1}^{n} \alpha_i \mathcal{K}(x_i, x) + b. \tag{9}$$

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_7

4 Implementation

In this section we describe the used data sets, the QUBO formulation and the solvers that are used for benchmarking.

4.1 Data

Two public datasets have been used. First, a Standard Banknote Authentication[3] dataset has been used. We only considered two attributes: variance of the wavelet function and skewness of the Wavelet function. We also used the well known Iris[4] dataset, while only considering Iris-sentosa and Iris-versicolor for binary classification. The following 2 attributes have been used for classification here: sepal length and sepal width.

The data has been randomised before training, and each datapoint was scaled according to the following:

$$x_{\text{new}} = \frac{x - x_{\text{min}}}{x_{\text{max}} - x_{\text{min}}}. \tag{10}$$

In our experiment, subsets of various sizes are taken for comparison of the scalability of the implementations. We use two thirds of the data points as training data and the remaining one third as our validation set.

4.2 QUBO formulation

To translate the quadratic programming formulation of Equations (4)-(6) to a QUBO formulation there are two main steps. First, the input has to be translated to binary input, using the encoding:

$$\alpha_n = \sum_{k=0}^{K-1} B^k a_{K_{n+k}}, \tag{11}$$

with $a_{K_{n+k}} \in \{0, 1\}$ binary variables, K the number of binary variables to encode α_n and B the base used for the encoding, usually $B = 2$. More details about the choice for K can be found in [27]. The second step is to translate the constraints (Eq. (5)-(6)) to the objective function (Eq 4), using a penalty factor ξ for a quadratic penalty. The value of ξ is determined before the training phase(if no particular value of ξ is known, a good strategy is to try exponentially growing sequences, $\xi = \{..., 10^{-4}, 10^{-3}, 10^{-2}, ...\}$) (We have optimized the value of ξ). The resulting objective function then becomes:

$$\frac{1}{2} \sum_{n,m,k,j} a_{K_{n+k}} a_{K_{m+j}} B^{k+j} y_n y_m \mathcal{K}(x_n, x_m) - \sum_{n,k} B_k a_{K_{n+k}} + \xi (\sum_{n,k} B^k a_{K_{n+k}} y_n)^2. \tag{12}$$

[3] http://archive.ics.uci.edu/ml/datasets/banknote+authentication
[4] http://archive.ics.uci.edu/ml/datasets/Iris/

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_7

4.3 Solvers

Four main implementations are used. A short description of all four are given here.

QPU Implementation The first implementation is directly on the D-Wave Quantum Processor Unit (QPU). First, a minor embedding has to be created, as described in Section 2. Next, due to the architecture of the quantum chip, there are limitations to the number of direct connections between qubits. While some qubits in the chip are connected using external couplers, the D-Wave QPU is not fully connected. Hence a problem variable has to be duplicated to multiple connected qubits. All these qubits representing the same variables are part of a so-called chain. For this, a penalty value (edge weights of the chain), called chain strength (λ), has to be found. In [6], Coffrin et al. provide a thorough analysis for its selection procedure. The DwaveSampler [10], automatically determines an initial value based on the QUBO at hand.

Hybrid Solvers For our second implementation, we use the Hybrid solvers offered by D-Wave Advantage [10], which implement state of the art classical algorithms with intelligent allocation of the QPU to parts of the problem where it benefits most, i.e., the sampler uses the energy impact as the criteria for selection of variables. These solvers are designed to accommodate even very large problems. This means that most parameters of the embedding are set automatically. By default, samples are iterated over four parallel solvers. The top branch implements a classical Tabu Search [13] that runs on the entire problem until interrupted by another branch completing. The other three branches use different decomposers to sample out parts of the current sample set and send it to different samplers. HQPU acts as a black box solver and the specific part of the problem which gets embedded on the QPU is unknown.

Simulated Annealing Simulated Annealing, is implemented using the SimulatedAnnealerSampler() sampler from the D-Wave Ocean Software Development Kit [9]. Simulated Annealing is a probabilistic method proposed by Kirkpatric, Gelett and Vecchi in [16] for finding the global minimum of a cost function. It is a meta-heuristic to approximate global optimisation in a large search space for optimisation problems. It works by emulating the physical process of first heating and then slowly cooling the system to decrease defects, thus minimizing energy. In each iteration the Simulated Annealing heuristic considers a neighbouring state s^* of the current state s and probabilistically decides whether to move the system to state s^* or not. Here, the probability distribution is based on a scale proportional to temperature. The heuristic accepts points that lower the objective, but also accepts points that raise the objective with a certain probability, hence avoiding being trapped in a local minimum. To converge the algorithm an annealing schedule is decided, which decreases the temperature as the heuristic proceeds. Lowering the temperature reduces the error probability,

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_7

and hence decreasing the extent of the search, which in turn leads the heuristic to converge to a global minimum.

Classical Implementation Finally, to implement the SVM's classically, we have used the scikit-learn [21] Python library. Scikit-learn uses LIBSVM [3] for implementing a support vector machine. The Quadratic formulation, Eq. (4), is formulated in its dual form:

$$\text{minimise } \frac{1}{2}\alpha^T Q\alpha - e^T\alpha, \tag{13}$$

subject to Eq. (5)-(6). Here $Q_{ij} = y_i y_j \mathcal{K}(x_i, x_j)$ and $e = [1, .., 1]$. Q is a dense positive semi-definite matrix and might become too large to store. To tackle this a decomposition method is used that modifies only a subset of Q. This subset B leads to a smaller optimisation problem. LIBSVM uses Sequential Minimal Optimisation [11], which restricts B to only two elements. Hence, a simple two variable problem is solved at each iteration without the need of any quadratic programming optimisation software like CPLEX[7] or Gurobi[14].

5 Results

We ran the simulations for the Standard Banknote Authentication dataset and the well known Iris Dataset, using the implementations discussed in subsection 4.3. The simulations were run using the following parameter values: $K = 2$, $B = 2$, $C = 3$, $\xi = 0.001$, $\gamma = 16$. We do not intend to optimize C, γ as our aim is to study the behaviour for same hyper parameters. However ξ has been optimized.

We use Key Performing Indicators (KPI), which are measurable values that depicts how effectively a classification model has performed. We have used Accuracy, $F1$-score, Precision and Recall as our KPI's. Accuracy is the fraction of samples that have been classified correctly, Precision is the proportion of correct positive identifications over all positive identifications, Recall is the proportion of correct positive identifications over all actual positives:

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}, \quad \text{Recall} = \frac{tp}{tp + fn}, \quad \text{Precision} = \frac{tp}{tp + fp},$$

where tp is true positive, fp is false positive, tn is true negative and fn is false negative. The $F1$-score is a way of combining the precision and recall of the model, and it is defined as the harmonic mean of the model's precision and recall:

$$F1 = \frac{tp}{tp + \frac{1}{2}(fp + fn)}.$$

The probability or the certainty with which a class is predicted by the model, is defined as the confidence of the classifier. The higher the absolute value of the

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_7

decision function, as shown in Equation (9), for a given data point, the more probable it is that the data point belongs to a particular class. Figures 1-6 represent the contour plot of the decision function, with the horizontal and vertical axis representing the data points and the decision function for the corresponding points being represented by the colour gradient.

In Table 1, we benchmark the results of the previously discussed implementations from Section 4.3 on two different randomized versions of the well known Iris Dataset while varying the size of the input dataset (N). For each instance we show the accuracy, $F1$-score and Lagrangian value(-Energy) for the best solution found by the solvers. We see that the hybrid solver (HQPU) and Simulated Annealing (SA) produce the same classifiers for all instances of the data. In our previous research we have seen that the hybrid solver outperforms SA while scaling for larger datasets [22].

In Table 2, we show the results of the previously discussed implementations on the Standard Banknode Authentication dataset. Similar to Table 1, the HQPU and SA produced identical classifiers. The HQPU and SA give reasonable results in comparison to the scikit-learn (Classical Implementation). We observe that the QPU is not able to find embeddings for instances larger that 90. QPU produces higher energy, inefficient solutions in comparison to HQPU and SA. From Figures 2 and 3, we observe that the most efficient solutions in terms of energy (produced by the QUBO formulation), have sparsely situated dark regions instead of a smoother dark region. These dark regions are concentrated around the support vectors. A smoother plot is obtained for the QPU simulation (Figure 1), which is credited to the inability of the solver to find the minimum energy solution. The solution obtained by the QPU has already been disregarded and improved by the HQPU and SA solvers. Unlike SA and HQPU, scikit-learn (Figure 4) produces smoother plots, i.e., plots with large dense regions. Hence, the classifier has a higher confidence when predicting data. Even though the hyperparamters are same, scikit-learn has a softer margin, i.e., it allows the SVM to make some mistakes while keeping the margin at the maximum so that other points can be classified correctly.

To test the error tolerance of the classifiers, we ran another simulation of the Banknote Authentication dataset, but with 4 artificially inserted data points. We inserted 2 Type 1 points in dark blue region and similarly, we inserted 2 Type -1 points in dark red region. The SA/HQPU (Figure 5) considered the artificially inserted datapoints as support vectors and created a small region around it giving a completely new classifier. The accuracy and $F1$ score showed a variation, and decreased from (0.94, 0.94) to (0.88, 0.89). The classifier ended up overfitting (hard margin) the data. This works extremely well for the training set, but not for the validation set, which can be seen from the decrease in the value of KPI's.

Although scikit-learn (Figure 6), did consider the points as support vectors, it doesn't create a new region for it, hence demonstrating a softer margin. The KPI's for this classifier also remained unaffected by the insertion of the new data.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_7

6 Conclusion

Quantum machine learning is still in its early stages and the implementation of machine learning paradigms on a quantum computer has shown immense capability. Quantum computing will provide the computational power needed when classical computers are reaching the upper cap of their performance, In this phase, hybrid solvers produce promising results by combining state of the art classical algorithms with a quantum annealer. In this paper, we compare the performance of hybrid solver, simulated annealing and direct QPU embedding for implementing the support vector machine. While comparing the sensitivity of the QUBO formulation in all the implementation to variation in input data.

With the limited development of the current hardware, the Quantum Processing Unit (QPU) is only able to solve small instances and is unable to find a suitable embedding for larger instances (> 90), for larger instances decomposition algorithms are required. Inefficient classifications are obtained by the QPU (for the small instances), which are outperformed by other implementations.

HQPU and SA have produced the same classifiers for all the input data, but from previous research we know that HQPU scales better. In Table 2, HQPU and SA have been able to outperform scikit-learn's implementation.
The scikit-learn implementation has produced higher confidence plots, while HQPU and SA have produced highly sparse plots centered around support vectors, which isn't desirable for the generalisation of a classifier. The QUBO formulations have been highly sensitive to small variations showing a hard margin for the same input parameters. A slight variation (2 misclassified in a data set consisting of 100 datapoints) lead to different classifiers. HQPU and SA were found to overfit the data for the same parameters as scikit-learn. The classical implementation on the other hand wasn't affected by a slight variation in the input dataset.

Future research comprises understanding some of the results further. First there is the sensitivity to small variations of the data. We want to investigate whether this comes from possible overfitting by the QUBO-based implementations, which follow from the various plots. Second, we saw a significant difference in Lagrangian value between the QUBO-based and the Scikit-Learn approaches. There might be other objective functions suitable for the QUBO formulation than currently used.

Acknowledgments

The authors thank Irina Chiscop and Tariq Bontekoe for their valuable review and comments.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_7

References

1. Bhatia, H.: Support vector machine implementation on d-wave quantum annealer, https://github.com/HarshilBhatia/-Support-Vector-Machine-Implementation-on-D-Wave-Quantum-Annealer
2. Booth, M., Reinhardt, S.P., Roy, A.: Partitioning Optimization Problems for Hybrid Classical/Quantum Execution. Tech. rep., D-Wave Systems (09 2017)
3. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2**, 27:1–27:27 (2011), software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
4. Chatterjee, R., Yu, T.: Generalized coherent states, reproducing kernels, and quantum support vector machines. Quantum Information & Computation **17**(15-16), 1292–1306 (2017)
5. Choi, V.: Minor-embedding in adiabatic quantum computation: I. the parameter setting problem (2008)
6. Coffrin, C.J.: Challenges with chains: Testing the limits of a d-wave quantum annealer for discrete optimization. Tech. rep., Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (2019)
7. Cplex, I.I.: V12. 1: User's manual for cplex. International Business Machines Corporation **46**(53), 157 (2009)
8. D-Wave-Systems: D-wave hybrid solver service: An overview, https://www.dwavesys.com/sites/default/files/14-1039A-A_D-Wave_Hybrid_Solver_Service_An_Overview.pdf
9. D-Wave-Systems: D-wave ocean sdk, https://github.com/dwavesystems/dwave-ocean-sdk
10. D-Wave-Systems: The d-wave advantage system: An overview (2020), https://www.dwavesys.com/sites/default/files/14-1049A-A_The_D-Wave_Advantage_System_An_Overview_0.pdf
11. Fan, R.E., Chen, P., Lin, C.: Working set selection using second order information for training support vector machines. J. Mach. Learn. Res. **6**, 1889–1918 (2005)
12. Glover, F., Kochenberger, G., Du, Y.: A tutorial on formulating and using qubo models. arXiv preprint arXiv:1811.11538 (2018)
13. Glover, F., Laguna, M.: Tabu Search, pp. 2093–2229. Springer US, Boston, MA (1998)
14. Gurobi Optimization, L.: Gurobi optimizer reference manual (2020), http://www.gurobi.com
15. Kadowaki, T., Nishimori, H.: Quantum annealing in the transverse ising model. Physical Review E **58**(5), 5355 (1998)
16. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. science **220**(4598), 671–680 (1983)
17. Li, R.Y., Di Felice, R., Rohs, R., Lidar, D.A.: Quantum annealing versus classical machine learning applied to a simplified computational biology problem. npj Quantum Information **4**(1) (2018)
18. Li, Z., Liu, X., Xu, N., Du, J.: Experimental realization of a quantum support vector machine. Physical Review Letters **114**(14) (2015)
19. Lucas, A.: Ising formulations of many np problems. Frontiers in Physics **2**, 5 (2014)
20. Neumann, N., Phillipson, F., Versluis, R.: Machine learning in the quantum era. Digitale Welt **3**(2), 24–29 (2019)
21. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A.,

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_7

Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
22. Phillipson, F., Bhatia, H.S.: Portfolio optimisation using the d-wave quantum annealer. In: International Conference on Computational Science. pp. 1–14. Springer (2021)
23. Phillipson, F., Wezeman, R.S., Chiscop, I.: Three quantum machine learning approaches or mobile user indoor-outdoor detection. In: 3rd International Conference on Machine Learning for Networking (2020)
24. Piattini, M., Peterssen, G., Pérez-Castillo, R., Hevia, J.L., Serrano, M.A., Hernández, G., de Guzmán, I.G.R., Paradela, C.A., Polo, M., Murina, E., et al.: The talavera manifesto for quantum software engineering and programming. In: QANSWER. pp. 1–5 (2020)
25. Rebentrost, P., Mohseni, M., Lloyd, S.: Quantum support vector machine for big data classification. Physical Review Letters **113**(13) (9 2014). https://doi.org/10.1103/physrevlett.113.130503, `http://dx.doi.org/10.1103/PhysRevLett.113.130503`
26. Wang, L.: Support vector machines: theory and applications, vol. 177. Springer Science & Business Media (2005)
27. Willsch, D., Willsch, M., De Raedt, H., Michielsen, K.: Support vector machines on the d-wave quantum annealer. Computer Physics Communications **248**, 107006 (2020)

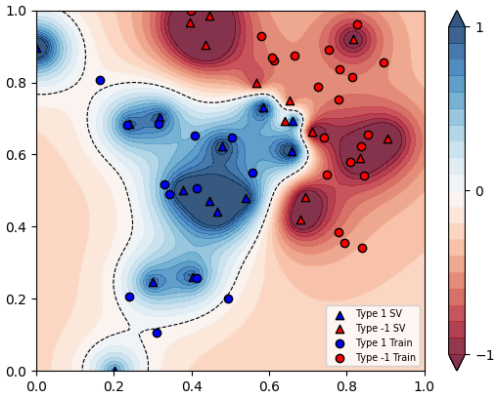

Fig. 1. Contour plot of decision function, support Vectors and input data points implemented using the QPU on the Standard Banknote Authentication Dataset with 60 samples.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_7

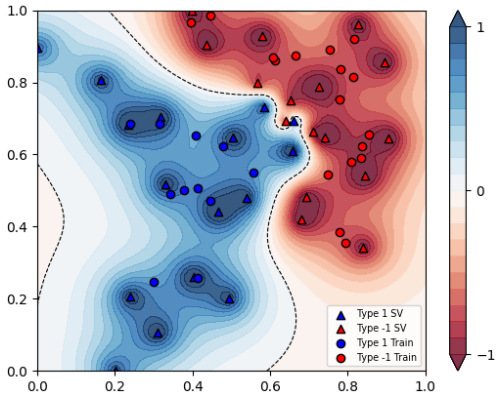

Fig. 2. Contour plot of decision function, support vectors and input data points implemented using the *HQPU and SA* solvers on the Standard Banknote Authentication Dataset with 60 samples.

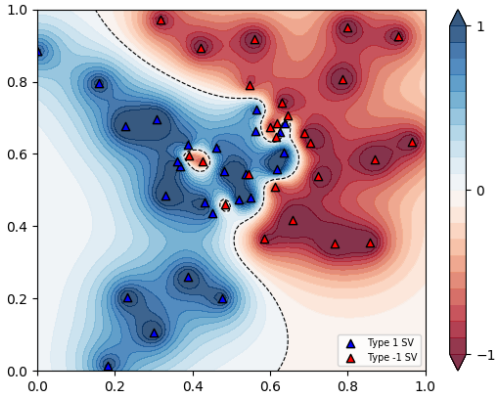

Fig. 3. Contour plot of decision function and support vectors implemented using the *HQPU and SA* solvers on the Standard Banknote Authentication Dataset with 60 samples.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_7

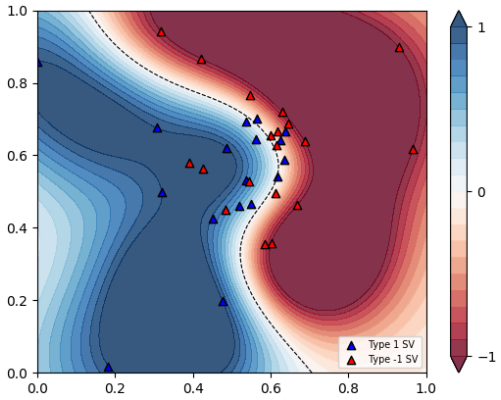

Fig. 4. Contour plot of decision function and support vectors implemented using the scikit-learn library on the Standard Banknote Authentication Dataset with 150 samples.

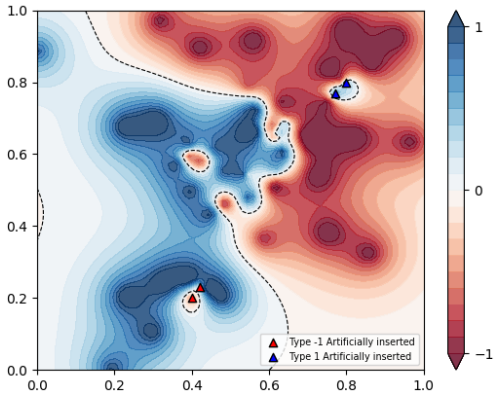

Fig. 5. Contour plot of decision function and artificially inserted data points using the HQPU and SA solvers on the Standard Banknote Authentication Dataset with 150 samples.

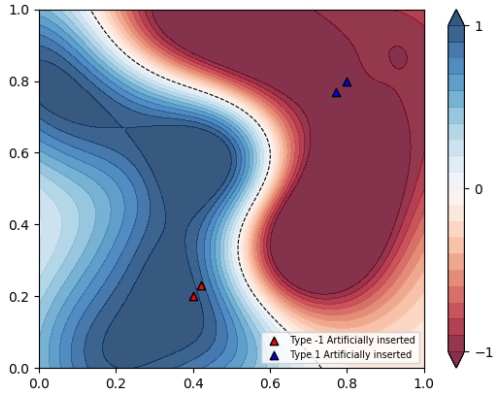

Fig. 6. Contour plot of decision function and artificially inserted data points using the scikit-learn library on the Standard Banknote Authentication Dataset with 150 samples.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_7

Table 1. Results for the Iris Dataset. Data 1 and Data 2 denote 2 different randomised version of the same.

	Data 1					Data 2				
Size	$F1$	Prec	Rec	Acc	Lagr	$F1$	Prec	Rec	Acc	Lagr
HQPU										
30	1	1	1	1	6.61	0.88	0.8	1	0.9	5.44
60	1	1	1	1	7.36	1	1	1	1	8.5
90	1	1	1	1	8.75	1	1	1	1	9.33
SA										
30	1	1	1	1	6.61	0.88	1	1	0.9	5.44
60	1	1	1	1	7.36	1	1	1	1	8.5
90	1	1	1	1	8.75	1	1	1	1	9.33
Scikit-learn										
30	1	1	1	1	4.32	0.88	1	1	0.9	4.03
60	0.96	0.9	1	0.96	3.16	1	0.9	1	1	2.17
90	1	1	1	1	3.13	1	1	1	1	2.06

Table 2. Results for the two randomized versions(denoted as Data1, Data2)of the Standard Banknote Authentication Dataset

	Data 1					Data 2				
Size	$F1$	Prec	Rec	Acc	Lagr	$F1$	Prec	Rec	Acc	Lagr
HQPU										
30	1	1	1	1	7.12	0.66	1	0.5	0.8	7.44
60	0.93	0.88	1	0.95	13.81	0.88	0.8	1	0.9	11.45
90	0.83	1	0.7	0.83	15.98	0.86	0.81	0.92	0.86	16.79
120	0.91	0.95	0.88	0.90	31.04	0.875	0.875	0.875	0.9	21.30
150	0.95	1	0.89	0.94	38.72	0.84	0.88	0.88	0.88	31.85
SA										
30	1	1	1	1	7.13	0.66	1	0.5	0.8	7.45
60	0.93	0.88	1	0.95	13.81	0.88	0.8	1	0.9	11.45
90	0.83	1	0.7	0.83	15.98	0.93	0.81	0.92	0.95	16.8
120	0.91	0.95	0.88	0.90	31.04	0.87	0.875	0.875	0.9	21.30
150	0.95	1	0.89	0.94	38.72	0.84	0.88	0.8	0.88	31.85
Scikit-Learn										
30	1	1	1	1	2.49	0.57	0.4	1	0.7	-4.45
60	1	1	1	1	5.99	0.88	1	0.8	0.9	-15.92
90	0.77	0.71	0.83	0.80	8.30	0.94	0.94	0.94	0.93	-21.18
120	0.91	0.9	0.9	0.9	13.43	0.85	0.82	0.88	0.88	-18.57
150	0.92	0.92	0.92	0.92	4.63	0.85	0.77	0.94	0.88	-34.23
QPU										
30	1	1	1	1	12	0.8	1	0.5	0.9	5.9
60	0.93	0.88	1	0.95	96.6	0.75	0.9	1	0.8	8.4
90	0.77	1	1	0.76	294	0.8	0.75	8	0.8	6.53

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77980-1_7