Assembly of a Coreset of Earth Observation Images on a Small Quantum Computer

Article *in* Electronics · October 2021

DOI: 10.3390/electronics10202482

2 authors:

Soronzonbold Otgonbaatar

9 PUBLICATIONS **7** CITATIONS

SEE PROFILE

Mihai Datcu

German Aerospace Center (DLR)

899 PUBLICATIONS **6,573** CITATIONS

SEE PROFILE

Assembly of a Coreset of Earth Observation Images on a Small Quantum Computer

Soronzonbold Otgonbaatar, Mihai Datcu, *Fellow, IEEE*

***Abstract*—Satellite instruments monitor the Earth's surface day and night, and as a result, the size of Earth observation (EO) data is dramatically increasing. Machine Learning/Deep Learning (ML/DL) techniques are employed routinely to analyze and process these big EO data, and one well-known ML technique is a Support Vector Machine (SVM). An SVM poses a quadratic programming problem, and quantum computers including quantum annealers (QA) as well as gate-based quantum computers promise to solve an SVM more efficiently than a conventional computer; training the SVM by employing a quantum computer/conventional computer represents a quantum SVM (qSVM)/classical SVM (cSVM) application. However, quantum computers cannot tackle many practical EO problems by using a qSVM due to their very low number of input qubits. Hence, we assembled a coreset ("core of a dataset") of given EO data for training a *weighted* SVM on a small quantum computer. The coreset is a small, representative weighted subset of an original dataset, and its performance can be analyzed by using the proposed *weighted* SVM on a small quantum computer in contrast to the original dataset. As practical data, we use *synthetic* data, *Iris* data, a *Hyperspectral Image* (HSI) of Indian Pine, and a *Polarimetric Synthetic Aperture Radar* (PolSAR) image of San Francisco. We measured the closeness between an original dataset and its coreset by employing a Kullback-Leibler (KL) divergence test, and in addition, we trained a *weighted* SVM on our coreset data by using both a D-Wave quantum annealer (D-Wave QA) and a conventional computer. Our findings show that the coreset approximates the original dataset with very small KL divergence, and the *weighted* qSVM even outperforms the *weighted* cSVM on the coresets for a few instances of our experiments. As a side result (or a by-product result), we also present our KL divergence findings for demonstrating the closeness between our original data (i.e., our synthetic data, Iris data, hyperspectral image, and PolSAR image) and the assembled coreset.**

***Index Terms*—coreset assembly, quantum Support Vector Machine, hyperspectral images, PolSAR images, quantum Machine Learning.**

I. Introduction

Remotely sensed images are used for EO and acquired by means of aircraft or satellite platforms. The acquired images from satellites are available in digital format and consist of information on the number of spectral bands, radiometric resolution, spatial resolution, etc. A typical EO dataset is big, massive, and hard to classify by using ML/DL techniques when compared with conventional non-satellite images [1], [2]. In principle, ML/DL techniques are a set of methods for recognizing and classifying common patterns in a labeled/unlabeled dataset [3], [4]. However, they are computationally expensive and intractable to train big massive data. Recently, several studies proposed to use only a coreset ("core of a dataset") of an original dataset for training ML/DL methods and tackling intractable posterior distributions via Bayesian inference [5], [6], [7], even for a Support Vector Machine (in short, SVM) [8]. The coreset is a small, representative weighted subset of an original dataset, and ML/DL methods trained on the coreset yield results being competitive with the ones trained on the original dataset.

The concept of a coreset opens a new paradigm for training ML/DL models by using small quantum computers [9], [10] since currently available quantum computers offered by *D-Wave Systems* (D-Wave QA) and by *IBM quantum experience* (a gate-based quantum computer) comprise a very few quantum bits (qubits) [11], [12]. In particular, quantum computers promise to solve some intractable problems in ML/DL [13], [14], [15], and to train an SVM even better/faster than a conventional computer when its input data volume is very small ("core of a dataset") [16], [17]. Training ML/DL methods by using a quantum computer or by exploiting quantum information is called Quantum Machine Learning (QML) [18], and finding the solutions of the SVM on a quantum computer is termed a quantum SVM (qSVM), otherwise classical SVM (cSVM).

This work uses a D-Wave QA for training a weighted SVM, and our method can be easily adapted and extended for a gate-based quantum computer. The D-Wave QA has a very small number of input qubits and a specific *Pegasus* topology for the connectivity of its qubits [19], and it is solely designed for solving a Quadratic Unconstrained Binary Optimization (QUBO) problem [14], [20]. For practical EO data, there is a benchmark and a demonstration example for training an SVM with binary quantum classifiers when using a D-Wave QA [21], [22]. Here, the SVM is a quadratic programming problem considered as a QUBO problem. Furthermore, there is a challenge to embed the variables of a given SVM problem into the *Pegasus* topology (i.e., the connectivity constraint of qubits), and to overcome this constraint of a D-Wave QA, the authors of [21] employed a *k-fold* approach to their EO data such that the size of variables in the SVM satisfies the connectivity constraint of qubits of a D-Wave QA.

In this article, we construct the coreset of an original dataset via *sparse variational inference* [6] and then employ this coreset for training the weighted SVM by using a D-Wave QA. Further, we train the weighted SVM, posed as a QUBO problem, by using a D-Wave QA on the coreset instead of the original massive data, and we benchmark our classification results with respect to the *weighted* cSVM. As for practical and real-world EO data, we use *synthetic* data, *Iris* data, a

Affiliations: German Aerospace Center (DLR), *Oberpfaffenhofen*
Emails: soronzonbold.otgonbaatar@dlr.de, mihai.datcu@dlr.de

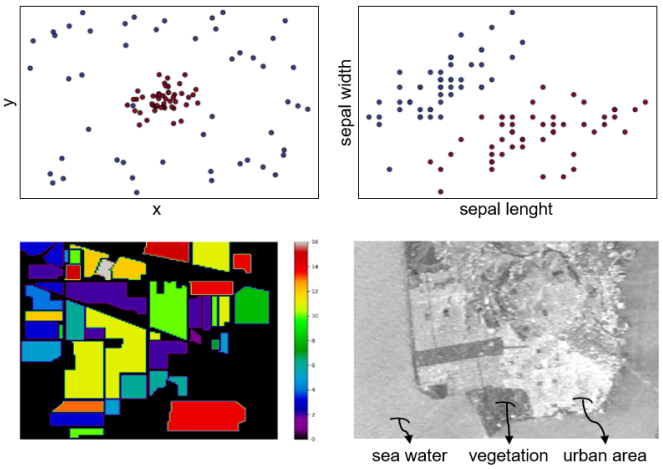

Fig. 1. Top (Left to Right): *Synthetic* data with two classes, and *Iris* data with two classes (Iris Setosa, and Iris Versicolour) charachterized by two features (sepal length, sepal width); Bottom (Left to Right): Indian Pine *HSI* with 16 classes {1: Alfalfa, 2: Corn-notill, 3: Corn-mintill, 4: Corn, 5: Grass-Pasture, 6: Grass-Trees, 7: Grass-Pasture-mowed, 8: Hay-windrowed, 9: Oats, 10: Soybean-notill, 11: Soybean-mintill, 12: Soybean-clean, 13: Wheat, 14: Woods, 15: Building-Grass-Drives, 16: Stones-Steel-Towers}, and *PolSAR* image of San Francisco with three classes.

Hyperspectral Image (HSI) of Indian Pine, and a *Polarimetric Synthetic Aperture Radar* (PolSAR) image of San Francisco charachterized by its Stokes parameters [23].

Our paper is structured as follows: In Section II, we present our datasets, and we construct the coresets of our datasets in Section III. We introduce a *weighted* cSVM, and construct a *weighted* qSVM for our experiments in Section IV. Then we train the *weighted* qSVM on our coresets by using a D-Wave QA, and demonstrate our results with respect to the *weighted* cSVM in Section V. Finally, we draw some conclusions in Section VI.

II. Our Datasets

We use four different datasets, namely *synthetic* data, *Iris* data, an Indian Pine *HSI*, and a *PolSAR* image of San Francisco characterized by its Stokes parameters [23], [24], [25], [26]. The first two sets are used to understand the concept of a coreset, and the implementation of a weighted SVM on their coresets by using a D-Wave QA. Namely, we use the coresets of the first two to set the internal parameters of a D-Wave QA since the solutions generated by the D-Wave QA are affected by those internal parameters (called *annealing parameters*). The last two sets are employed as real-world EO data for constructing their coresets and for training the *weighted* qSVM on their coresets after the annealing parameters are set in a prior (see Figure 1). In the next sections, we use a notation "*weighted qSVM*" meaning that "training a *weighted SVM* posed as a QUBO problem by using a D-Wave QA".

A. Synthetic data

We generated *synthetic* data with two classes $(\mathbf{x}_n, y_n)$ according to

TABLE I
The two classes of Synthetic and Iris data

	Synthetic data	Iris data
Classes	$\{-1, +1\}$	{setosa, versicolour}
Data size	100	100

TABLE II
The two classes of the Indian Pine HSI; $\{1, 2\}$ represents {Alfalfa, Corn-notill}, $\{2, 3\}$ represents {Corn-notill, Corn-mintill}, etc.

Indian Pine HSI						
Classes	$\{1, 2\}$	$\{2, 3\}$	$\{3, 4\}$	$\{4, 5\}$	$\{5, 6\}$	$\{6, 7\}$
Data size	295	452	214	144	243	758

TABLE III
The two classes of our PolSAR image

PolSAR image of San Francisco		
Classes	{urban area, sea water}	{vegetation, sea water}
Data size	61,465	61,465

$$\mathbf{x}_n = r_n \begin{pmatrix} \cos\phi_n \\ \sin\phi_n \end{pmatrix} + \begin{pmatrix} \epsilon_n^x \\ \epsilon_n^y \end{pmatrix}, \quad y_n \in \{-1, +1\}, \tag{1}$$

where $r_n = 1$ if $y_n = -1$, and $r_n = 0.15$ if $y_n = +1$. ϕ_n is linearly spaced in $(0, 2\pi]$ for each class, and $\epsilon_n^x, \epsilon_n^y$ are samples drawn from a normal distribution with $\mu = 0, \sigma = 1$. We are replicating the data already demonstrated for training an SVM by using a D-Wave QA described in [27]. Moreover, we have $(\mathbf{x}_n, y_n), n = 1, \ldots, 100$ data points shown in Figure 1 Top (Left) and in Table I.

B. Iris data

Iris data consist of three classes (Iris Setosa, Iris Versicolour, and Iris Virginica), each of which has four features, namely sepal length, sepal width, petal length, and petal width [24]. We consider a two-class dataset {Iris Setosa, Iris Versicolour} with a size of 100 data points, and each class is characterized by two features (sepal length, sepal width) shown in Figure 1 Top (Right) and Table I.

C. Indian Pine HSI

An Indian Pine HSI obtained by AVIRIS sensor has 16 classes, and each class is charachterized by 200 spectral bands [25] (see Fig. 1 Bottom (Left)). For simplicity, we use only two-classes (see Table II), and each class is charachterized by two features instead of 200 spectral bands by exploiting Principal Component Analysis (PCA) [22].

D. PolSAR image of San Francisco

Each pixel of our *PolSAR* image is characterized by a 2×2 scattering matrix as follows

$$S = \begin{pmatrix} s_{HH} & s_{HV} \\ s_{VH} & s_{VV} \end{pmatrix}, \tag{2}$$

where the first index of $s_{ij}, i, j \in \{H, V\}$ represents the polarization state of the incident polarized beam, and its second

index represents the polarization state of the reflected polarized beam on targets [28], [29]. The off-diagonal elements of S are equal $s_{VH} = s_{HV}$ since our *PolSAR* image of San Francisco is a fully-polarized PolSAR image obtained by a monostatic radar.

The incident/reflected polarized beam can be represented by its complex amplitude in a polarization basis $\{\hat{H}, \hat{V}\}$ by

$$\vec{E}_0 = E_{H0}\hat{H} + E_{V0}\hat{V}. \tag{3}$$

The complex amplitude vector can be expressed by a so-called *Jones vector*

$$\vec{J} = \begin{pmatrix} E_{H0} \\ E_{V0} \end{pmatrix} = \begin{pmatrix} |E_{H0}|e^{i\phi_H} \\ |E_{V0}|e^{i\phi_V} \end{pmatrix}. \tag{4}$$

where ϕ_i are the phases of the polarized states. Further, the scattering matrix S expressed in (2) is a mapping such that

$$S : \vec{J}_i \to \vec{J}_r, \quad \vec{J}_r = S\vec{J}_i, \tag{5}$$

where $\vec{J}_i, \vec{J}_r$ is an incident and a reflected Jones vector, respectively. In matrix form, (5) can be re-expressed as

$$\begin{pmatrix} E^r_{H0} \\ E^r_{V0} \end{pmatrix} = \begin{pmatrix} s_{HH} & s_{HV} \\ s_{VH} & s_{VV} \end{pmatrix} \begin{pmatrix} E^i_{H0} \\ E^i_{V0} \end{pmatrix}. \tag{6}$$

The intensity of the reflected Jones vector is defined by

$$J = \begin{pmatrix} \langle E^r_{H0}E^{r*}_{H0}\rangle & \langle E^r_{H0}E^{r*}_{V0}\rangle \\ \langle E^r_{V0}E^{r*}_{H0}\rangle & \langle E^r_{V0}E^{r*}_{V0}\rangle \end{pmatrix} = \begin{pmatrix} J_{HH} & J_{HV} \\ J_{VH} & J_{VV} \end{pmatrix} \tag{7}$$

where $\langle\cdot\rangle$ stands for spatial averaging with a window size 7×7 pixels, and * for conjugation. Furthermore, we can re-express this intensity by

$$\begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} J_{HH} + J_{VV} \\ J_{HH} - J_{VV} \\ J_{VH} + J_{HV} \\ i(J_{HV} - J_{VH}) \end{pmatrix}, \tag{8}$$

where q_1, q_2, and q_3 are called *Stokes vectors*. We normalize these *Stokes vectors* according to

$$q_1 = \frac{q_1}{q_0}, \quad q_2 = \frac{q_2}{q_0}, \quad q_3 = \frac{q_3}{q_0}, \tag{9}$$

and the normalized q_1, q_2, and q_3 are called *Stokes parameters* [23].

Moreover, in this study, we use two classes for our PolSAR image of San Francisco, and the two classes are {urban area, sea water}, and {vegetation, sea water} shown in Figure 1 Bottom (Right) and in Table III. In addition, each class is characterized by its Stokes parameters (q_1, q_2, q_3) defined in (9).

III. Coresets of our datasets

In Bayesian inference, a posterior density $p(\theta|x)$ is written for θ parameters and for $\{(x_i, t_i)\}_{i=1}^N$ data points with its labels t_i by

$$p(\theta|x) = \frac{1}{Z}\exp\left\{\sum_{i=1}^{N} f_i(\theta)\right\}p_0(\theta), \tag{10}$$

TABLE IV
Coresets of our datasets presented in Table I-III, and the closeness between the original dataset and its coreset is measured by KL divergence.

Classes	Data size	Coreset Size	KL divergence
$\{-1, +1\}$	100	20	0.008194
{setosa, versicolour}	100	22	0.053002
{1, 2}	295	79	0.573451
{2, 3}	452	56	0.003121
{3, 4}	214	33	0.000600
{4, 5}	144	41	0.017201
{5, 6}	243	41	0.001823
{6, 7}	758	125	0.492636
{urban area, sea water}	61,465	501	0.125072
{vegetation, sea water}	61,465	343	0.272749

where Z is a partition function, $f_i(\theta)$ is a potential function, and $p_0(\theta)$ is a prior. For big massive data, the estimation of the posterior distribution is intractable, and hence, in practice, a Markov Chain Monte Carlo (MCMC) method is widely used to obtain samples from the posterior expressed by (10) [30].

To reduce the computational time of an MCMC method, the authors of [5]-[7] proposed to run the MCMC method on a small, weighted subset (i.e., coreset) of big massive data. They derived a sparse vector of nonnegative weights w such that only $M \ll N$ are non-zero, where M is the size of a coreset. Further, the authors proposed to approximate the weighted posterior distribution and run the MCMC method on the approximate distribution as follows:

$$p_w = p_w(\theta|x) = \frac{1}{Z(w)}\exp\left\{\sum_{i=1}^{N} w_i f_i(\theta)\right\}p_0(\theta). \tag{11}$$

We denote the full distribution of an original big massive dataset as $p_1 = p_1(\theta|x)$. More importantly, this posterior becomes tractable.

For assembling the coresets of our datasets presented in Table I-III, we use an algorithm via *sparse variational inference* for finding the sparse vector of nonnegative weights w and for approximating the posterior distribution (11) proposed by [6]. Here, the sparse vector of nonnegative weights w is found by optimizing a *sparse variational inference* problem:

$$\hat{w} = \min_{w} D_{KL}(p_w||p_1) \quad s.t. \quad w \geq 0, \quad \|w\|_0 \leq M, \tag{12}$$

where $\hat{w}$ is an optimal sparse vector weight, and $D_{KL}(p_w||p_1)$ is the Kullback-Leibler (KL) divergence which measures the similarity between two distributions (smaller is better):

$$D_{KL}(p_w||p_1) = \sum_x p_w \log\frac{p_w}{p_1}. \tag{13}$$

Moreover, the smaller value of the KL divergence test implies that we estimate the parameters θ in (11) by using the assembled coreset yielding similar results in comparison to the ones in (10) by using its original massive dataset. The used code is available in [31].

We derived the optimal sparse vector weights $\hat{w}$ and the coreset of our dataset such that

$$\{(x_i, t_i)\}_{i=1}^{N} \quad \rightarrow \quad \{(c_i, t_i, \hat{w}_i)\}_{i=1}^{M}, \quad \hat{w}_i \in R_{\geq 0}, \tag{14}$$

where (x_i, t_i) represents an original dataset, while $(c_i, t_i, \hat{w}_i)$ is our newly assembled coreset. In addition, we computed the similarity between our datasets and the corresponding coresets by using their KL divergences (see Table IV). Our results show that our coresets are very small in size compared with our original datasets, and the KL divergences between the original dataset and our coresets is comparatively small in most instances.

IV. Weighted Classical and Quantum SVMs on our Coresets

A. Weighted classical SVMs

In the previous section, we assembled the coreset of our original datasets shown in Table IV as

$$\{(c_i, t_i, \hat{w}_i)\}_{i=1}^{M}, \quad c_i \in R^2, \quad \hat{w}_i \in R_{\geq 0}. \tag{15}$$

To train a weighted SVM for our coresets represented via (15) by using a conventional computer, we express a weighted SVM as

$$\begin{aligned} &\textit{minimize} \quad H(\alpha) = \frac{1}{2}\sum_{ij} \alpha_i \alpha_j t_i t_j k(c_i, c_j) - \sum_i \alpha_i \\ &\textit{subject to} \quad 0 \leq \alpha_i \leq C_i, \\ &\textit{and} \quad \sum_i \alpha_i t_i = 0, \quad \alpha_i \in R, \end{aligned} \tag{16}$$

where $C_i = \hat{w}_i C$ is a regularization parameter, and $k(\cdot, \cdot)$ is the kernel function of the SVM [30]. This formulation of the SVM is called a *weighted* cSVM [32]; sometimes it is called a kernel-based *weighted* cSVM. The point c_i with $\alpha_i \neq 0$ is called a support vector.

After training the *weighted* cSVM, for a given test point $x_t \in R^2$, the decision function for its class label is defined by:

$$\hat{t} = sign\left[f(x_t)\right] = sign\left[\sum_i \alpha_i t_i k(c_i, x_t) + b\right], \tag{17}$$

where $sign(f(x_t)) = 1$ if $f(x_t) > 0$, $sign(f(x_t)) = -1$ if $sign(f(x_t)) < 0$, and $sign(f(x_t)) = 0$ otherwise. The decision boundary is an optimum hyperplane drawn by data points such that $f(x_t) = 0$ [30]. The bias b is expressed following [27]:

$$b = \frac{\sum_i \alpha_i (C_i - \alpha_i)\left[t_i - \sum_j \alpha_j t_j k(c_j, c_i)\right]}{\sum_i \alpha_i (C_i - \alpha_i)}. \tag{18}$$

The kernel-based *weighted* cSVM is a powerful technique since the kernel function maps non-separable features to higher dimensional separable features, and the decision boundary is less sensitive to outliers due to the weighted constraints C_i [27], [32]. Furthermore, the choice of the kernel function has a huge impact on the decision boundary, and the types of the kernel function are linear, polynomial, Matern, and a radial basis function (rfb) [30]. A widely-used kernel is an rbf defined by

$$\textit{rbf}(c_i, c_j) = \exp\Big\{-\gamma \|c_i - c_j\|^2\Big\}, \tag{19}$$

where $\gamma > 0$ is a parameter.

B. Weighted quantum SVMs

A weighted quantum SVM (in short, *weighted* qSVM) is the training result of the *weighted* cSVM given in (16) on a D-Wave QA. The D-Wave QA is a quantum annealer with a specific *Pegasus* topology for the interaction of its qubits, and it is specially designed to solve a QUBO problem:

$$H(\mathbf{z}) = \sum_{i,j} z_i Q_{ij} z_j, \quad z_i, z_j \in \{0, 1\}, \tag{20}$$

where z_i, z_j are called logical variables, and Q_{ij} includes a bias term h_i and the interaction strength of the logical variables g_{ij} [19]. Physical states of the Pegasus topology are called physical variables, two-state qubits residing at the edges of the Pegasus topology; a QUBO problem is also called a *problem energy*. The D-Wave QA anneals (evolves) from an initial to its final energy (problem energy) according to

$$H(T) = (1 - \varepsilon(T))H_0 + \varepsilon(T)H(\mathbf{z}), \tag{21}$$

where H_0 is an initial energy, T is the annealing time in microseconds, and $\varepsilon(T)$ is an annealing parameter in the range of $[0, 1]$.

Furthermore, to train the *weighted* qSVM on our coresets by using a D-Wave QA, we pose the *weighted* cSVM with a rbf kernel expressed by (16) and (19) as a QUBO problem. Here, we duplicate the formulation for posing the *weighted* cSVM as a QUBO problem in the article [27].

The variables of the *weighted* cSVM are decimal integers when the ones of the QUBO problem are binaries. Hence, we use a one-hot encoding form for the variables of the *weighted* cSVM

$$\alpha_i = \sum_{k=0}^{K-1} B^k z_{Ki+k}, \quad z_{Ki+k} \in \{0, +1\} \tag{22}$$

where K is the number of binary variables (bits), and B is the base. We insert this one-hot encoding form into the *weighted* cSVM given in (16), and formulate the second constraint of (16) as a squared penalty term

$$\left(\sum_i \alpha_i t_i\right)^2 = 0 \quad \rightarrow \quad \left(\sum_{ik} B^k z_{Ki+k} t_i\right)^2 = 0. \tag{23}$$

By using a Lagrange multiplier λ, we transform our *weighted* cSVM into the QUBO problem (20)

$$\begin{aligned} \textit{minimize} \quad H(z) &= \frac{1}{2}\sum_{ijkl} z_{Ki+k} z_{Kj+l} B^{k+l} t_i t_j k(c_i, c_j) \\ &\quad - \sum_{ik} B^k z_{Ki+k} + \lambda \left(\sum_{ik} B^k z_{Ki+k} t_i \right)^2 \\ &= \sum_{ij} \sum_{kl} z_{Ki+k} Q_{Ki+k,Kj+l} z_{Kj+l}, \end{aligned} \tag{24}$$

where

$$Q_{Ki+k,Kj+l} = \frac{1}{2} B^{k+l} t_i t_j (k(c_i, c_j) + \lambda) - \delta_{ij}\delta_{kl} B^k. \tag{25}$$

Note that the first constraint of (16) is satisfied automatically since the one-hot encoding form given in (22) is always greater than zero, and hence, the maximum value for each α_i is given by

$$C_i = \hat{w}_i \sum_{k=1}^{K} B^k. \tag{26}$$

For training the *weighted* qSVM, we concentrated on four hyperparameters which are the parameter γ of the RBF expressed by (19), the number of binary bits K, the base B, and the Lagrange multiplier λ given in (24); thus, we used the hyperparameters (γ, K, B, λ). For our applications, we set these hyperparameters to $(\gamma = 16, K = 2, B = 2, \lambda = 0)$ as proposed by [27] since these settings of the hyperparameters for the *weighted* qSVM generate competitive results with the ones generated by the *weighted* cSVM. For setting the bias defined in (18), we used the *weighted* cSVM.

In the next section, we train the *weighted* cSVM given in (16) and the *weighted* qSVM expressed by (24) on our coresets (see Table IV). In addition, we demonstrate how to program a D-Wave QA for obtaining a good solution of (24) since the solutions yielded by a D-Wave QA are greatly dependent upon the *embedding* of the logical variables into their corresponding physical variables, and the *annealing parameters* (annealing time, number of reads, and chain strength) [33].

V. Our Experiments

In our experiments, we trained our *weighted* cSVM and our *weighted* qSVM (models) on the coresets, and we tested our models on the original datasets (see Table IV). In addition, we set the hyperparameters of our *weighted* qSVM to $(\gamma = 16, K = 2, B = 2, \lambda = 0)$, and for training (i.e., for setting of the bias) of the *weighted* cSVM, we used the Python module *scikit-learn* [34].

For defining the *annealing parameters* (annealing time, number of reads, and chain strength) of a D-Wave QA, we first ran quantum experiments on *synthetic* two-class data, and *Iris* data. Then by leveraging these *annealing parameters*, we used our real-world EO data (the Indian Pine HSI and the PolSAR image of San Francisco) for evaluating our proposed method, "by training the *weighted* qSVM on the coreset of our EO data instead of a massive original EO data due to the small quantum computer (D-Wave QA) with only few qubits".

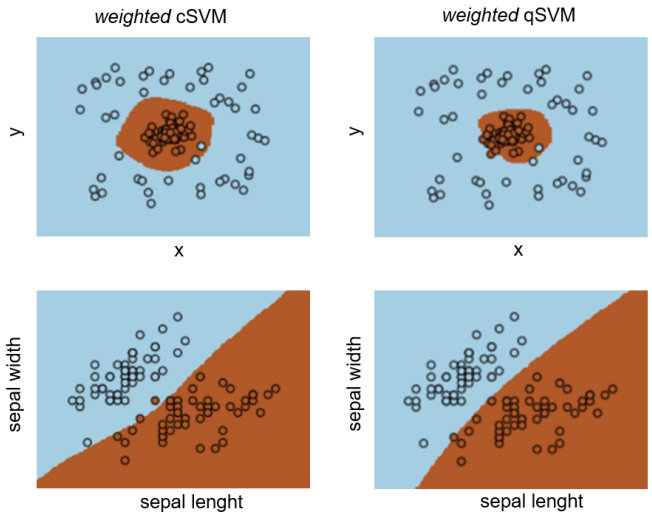

Fig. 2. Top: *Synthetic two data*; Bottom: *Iris* data. The visual results of our experiments generated by the *weighted* cSVM given in (16) and *weighted* qSVM expressed by (24). Our visual result demonstrates that our *weighted* qSVM generalizes the decision boundary of a given dataset better than its counterpart *weighted* cSVM.

A. Synthetic two-class data, and Iris data

For training the *weighted* qSVM expressed by (24), we first experimented on our coresets of *synthetic* two-class data and *Iris* data shown in Table IV in order to optimize the *annealing parameters* (annealing time, number of reads, and chain strength) of a D-Wave QA. In addition, we benchmarked the classification results generated by the *weighted* qSVM compared with the *weighted* cSVM. This had the advantage that we could easily tune the *annealing parameters* and visualize the generated results, both for quantum and classic settings.

In Figure 2 (in Table V), we show our results for the classification of *synthetic* two-class data and *Iris* data. Our results demonstrate that the *weighted* qSVM performs well in comparison with the *weighted* cSVM for both coresets (often better for *Iris* data).

To obtain these good solutions generated by our *weighted* qSVM, we set the annealing parameters of the D-Wave QA as follows:

- Annealing time: We controlled the annealing time by an *anneal schedule*. The anneal schedule is defined by the four series of pairs $[T, \varepsilon(T)]$ defined in (21). We set the annealing schedule accordingly: $[T, \varepsilon(T)] \in \{[0.0, 0.0], [1.0, 0.40], [19.0, 0.40], [20.0, 1.0]\}$.
- Number of reads: 10000
- Chain strength: 50.

B. Indian Pine HSI, and PolSAR image of San Francisco

As real-world EO data, we used the coresets of an Indian Pine HSI, and a PolSAR image of San Francisco for training the *weighted* qSVM when setting the annealing parameters of a D-Wave QA set as described above. Initially, we ran a number of quantum experiments on our coresets. In Table V we show the classification accuracy of our *weighted* qSVM

TABLE V
THE CLASSIFICATION ACCURACY OF THE WEIGHTED QUANTUM SVM (IN SHORT, QACC), AND THE WEIGHTED CLASSICAL SVM (IN SHORT, CACC) ON OUR CORESETS.

Classes	Coreset Size	QACC	CACC
$\{-1, +1\}$	20	0.95	0.97
{setosa, versicolour}	22	0.99	0.98
{1, 2}	79	0.96	0.96
{2, 3}	56	0.70	0.70
{3, 4}	33	0.88	0.88
{4, 5}	41	0.78	0.78
{5, 6}	41	0.71	0.71
{6, 7}	125	0.92	0.90
{urban area, sea water}	501	0.99	0.98
{vegetation, sea water}	343	0.99	0.99

results in comparison with the ones yielded by the *weighted* cSVM.

Our results explicitly demonstrate that the coresets obtained via *sparse variational inference* are small and representative subsets of our original datasets validated by their KL divergences shown in Table IV. In addition, our *weighted* qSVM generates its decision results with the same classification accuracy as for the *weighted* cSVM; in some instances, the *weighted* qSVM outperforms the *weighted* cSVM. Furthermore, by exploiting the coresets, we reduced the computational time of training with the *weighted* qSVM and the MCMC method for inferring the parameters of the posterior distribution as *proved* theoretically and *demonstrated* experimentally in [5] and [6].

VI. Discussion

Quantum algorithms (e.g., Grover's search algorithm) are designed to process data in quantum computers, and they are known to achieve quantum advantages over their conventional counterparts. Motivated by these quantum advantages, quantum computers based on quantum information science are being built for solving some problems (or running some algorithms) more efficiently than a conventional computer. However, currently available quantum computers (a D-Wave quantum annealer, and a gate-based quantum computer) are very small in input quantum bits (qubits). A very specific type of a quantum computer is a D-Wave quantum annealer (QA); it is designed to solve a Quadratic Unconstrained Binary Optimization (QUBO) problem belonging to a family of quadratic programming problems better than conventional methods.

For Earth observation, satellite images obtained from aircraft or satellite platforms are massive and represent hard heterogeneous data to train ML/DL models on a conventional computer. As a practical and real-world EO dataset, we used *synthetic* data, *Iris* data, a *Hyperspectral Image* (HSI) of Indian Pine, and a *Polarimetric Synthetic Aperture Radar* (PolSAR) image of San Francisco. One of well-known methods in ML/DL is a Support Vector Machine (SVM): This represents a quadratic programming problem. A global minimum of such a problem can be found by employing a classical method. However, its quadratic form allows us to use a D-Wave QA for finding the solution of an SVM better than a conventional computer. Thus, we can pose an SVM as a QUBO problem, and we named an SVM-to-QUBO transformation as a quantum SVM (qSVM). Then we can train the qSVM on our real-world EO data by using a D-Wave QA. However, the number of the physical variables of the qSVM is much larger than that of the logical variables of a D-Wave QA due to the massive EO data and the very few qubits.

Therefore, in our paper, we employed the *coreset* ("core of a dataset") concept via *sparse variational inference*, where the coreset is a very small and representative weighted subset of the original dataset. By assembling and exploiting the coreset of *synthetic* data and *Iris* data shown in Table IV, we trained a *weighted* qSVM posed as a QUBO problem on these coresets in order to set the *annealing parameters* of a D-Wave QA. We then presented our obtained visual results and the classification accuracy of *synthetic* and *Iris* data in Figure 1 and in Table V, respectively, in contrast to the ones of the *weighted* cSVM. Our results show that the *weighted* qSVM is competitive in comparison with the *weighted* cSVM $-$ and for *Iris* data even better than the *weighted* cSVM.

Finally, we assembled the coresets of our real-world EO data (from an *HSI* of Indian Pine, and a *PolSAR* image of San Francisco), and demonstrated the similarity between our real-world EO data and its coreset by analyzing their KL divergence. The KL divergence test proved that our coresets are valid small and representative weighted subsets of our real-world EO data (see Table IV). Then we trained the *weighted* qSVM on our coresets by using a D-Wave QA to prove that our *weighted* qSVM generates classification results being competitive with the *weighted* cSVM in Table V. The annealing parameters of the D-Wave QA were already defined in the prior section. In some instances, one can see that our *weighted* qSVM outperforms the *weighted* cSVM.

As an ongoing and a future work, we intend to develop a novel method for assembling coresets with balanced labels via *sparse variational inference* since currently available techniques generate unbalanced labels. Further, we plan to design hybrid quantum-classical methods for different real-world EO problems. These hybrid quantum-classical methods will perform a dimensionality-reduction of remotely-sensed images (in the spatial-dimension) by using the established methods, and will reduce the size of our training/test data by using a coreset generating balanced labels when we process these small datasets on a small quantum computer.

ACKNOWLEDGMENT

We would like to greatly acknowledge Gottfried Schwarz (DLR, Oberpfaffenhofen) for his valuable comments and contributions for enhancing a quality of our paper. Also, the authors gratefully acknowledge the Juelich Supercomputing Centre (https://www.fzjuelich.de/ias/jsc) for funding this project by providing computing time through the Juelich UNified Infrastructure for Quantum computing (JUNIQ) on a D-Wave quantum annealer.

REFERENCES

[1] J. Richards, "Remote sensing digital image analysis," *Berlin*, 10 2013.

[2] G. Cheng, X. Xie, J. Han, L. Guo, and G.-S. Xia, "Remote sensing image scene classification meets deep learning: Challenges, methods, benchmarks, and opportunities," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 3735–3756, 2020.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available: https://doi.org/10.1038/nature14539

[4] D. Marmanis, M. Datcu, T. Esch, and U. Stilla, "Deep learning earth observation classification using imagenet pretrained networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 1, pp. 105–109, 2016.

[5] D. Manousakas, Z. Xu, C. Mascolo, and T. Campbell, "Bayesian pseudocoresets," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 14 950–14 960. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/ab452534c5ce28c4fbb0e102d4a4fb2e-Paper.pdf

[6] T. Campbell and B. Beronov, "Sparse variational inference: Bayesian coresets from scratch," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/7bec7e63a493e2d61891b1e4051ef75a-Paper.pdf

[7] T. Campbell and T. Broderick, "Bayesian coreset construction via greedy iterative geodesic ascent," *ArXiv*, May 2018. [Online]. Available: https://arxiv.org/abs/1802.01737

[8] M. Tukan, C. Baykal, D. Feldman, and D. Rus, "On coresets for support vector machines," *ArXiv*, Feb 2020. [Online]. Available: https://arxiv.org/abs/2002.06469

[9] A. W. Harrow, "Small quantum computers and large classical data sets," *ArXiv*, Mar 2020. [Online]. Available: https://arxiv.org/abs/2004.00026

[10] T. Tomesh, P. Gokhale, E. R. Anschuetz, and F. T. Chong, "Coreset clustering on small quantum computers," *Electronics*, vol. 10, no. 14, 2021. [Online]. Available: https://www.mdpi.com/2079-9292/10/14/1690

[11] (2021) A D-Wave quantum annealer. [Online]. Available: https://cloud.dwavesys.com/leap

[12] (2021) IBM quantum experience. [Online]. Available: https://quantum-computing.ibm.com/

[13] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, Aug 2018. [Online]. Available: http://dx.doi.org/10.22331/q-2018-08-06-79

[14] V. S. Denchev, S. Boixo, S. V. Isakov, N. Ding, R. Babbush, V. Smelyanskiy, J. Martinis, and H. Neven, "What is the computational value of finite-range tunneling?" *Phys. Rev. X*, vol. 6, p. 031015, Aug 2016. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.6.031015

[15] V. Dunjko and H. J. Briegel, "Machine learning & artificial intelligence in the quantum domain: a review of recent progress," *Reports on Progress in Physics*, vol. 81, no. 7, p. 074001, jun 2018. [Online]. Available: https://doi.org/10.1088/1361-6633/aab406

[16] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Phys. Rev. Lett.*, vol. 113, p. 130503, Sep 2014. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.113.130503

[17] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean, "Power of data in quantum machine learning," *Nature Communications*, vol. 12, no. 1, p. 2631, May 2021. [Online]. Available: https://doi.org/10.1038/s41467-021-22539-9

[18] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, "Quantum machine learning," *Nature*, vol. 549, no. 7671, pp. 195–202, Sep 2017. [Online]. Available: https://doi.org/10.1038/nature23474

[19] K. Boothby, P. Bunyk, J. Raymond, and A. Roy, "Next-generation topology of d-wave quantum processors," *arXiv:2003.00133*, Feb 2020. [Online]. Available: https://arxiv.org/abs/2003.00133

[20] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "Quantum computation by adiabatic evolution," *arXiv:0001106*, Jan 2000. [Online]. Available: https://arxiv.org/abs/quant-ph/0001106

[21] G. Cavallaro, D. Willsch, M. Willsch, K. Michielsen, and M. Riedel, "Approaching remote sensing image classification with ensembles of support vector machines on the d-wave quantum annealer," in *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, 2020, pp. 1973–1976.

[22] S. Otgonbaatar and M. Datcu, "A quantum annealer for subset feature selection and the classification of hyperspectral images," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 7057–7065, 2021.

[23] F. Shang and A. Hirose, "Quaternion neural-network-based polsar land classification in poincare-sphere-parameter space," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 9, pp. 5693–5703, 2014.

[24] (2021) Iris data set. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/iris

[25] (2021) Airborne visible/infrared imaging spectrometer (aviris). [Online]. Available: https://aviris.jpl.nasa.gov/

[26] S. R. Cloude and K. P. Papathanassiou, "Polarimetric sar interferometry," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 36, no. 5, pp. 1551–1565, 1998.

[27] D. Willsch, M. Willsch, H. De Raedt, and K. Michielsen, "Support vector machines on the d-wave quantum annealer," *Computer Physics Communications*, vol. 248, p. 107006, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S001046551930342X

[28] J.-S. Lee, W.-M. Boerner, D. Schuler, T. Ainsworth, I. Hajnsek, K. Papathanassiou, and E. Lüneburg, "A review of polarimetric sar algorithms and their applications," *Journal of Photogrammetry and Remote Sensing, China*, vol. 9, pp. 31–80, 01 2004.

[29] A. Moreira, P. Prats-Iraola, M. Younis, G. Krieger, I. Hajnsek, and K. Papathanassiou, "A tutorial on synthetic aperture radar," *IEEE Geoscience and Remote Sensing Magazine (GRSM)*, vol. 1, pp. 6–43, 03 2013.

[30] M. Sheykhmousa, M. Mahdianpari, H. Ghanbari, F. Mohammadimanesh, P. Ghamisi, and S. Homayouni, "Support vector machine versus random forest for remote sensing image classification: A meta-analysis and systematic review," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 6308–6325, 2020.

[31] (2021) Bayesian coresets: Automated, scalable inference. [Online]. Available: https://github.com/trevorcampbell/bayesian-coresets

[32] X. Yang, Q. Song, and A. Cao, "Weighted support vector machine for data classification," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, vol. 2, 2005, pp. 859–864 vol. 2.

[33] S. Otgonbaatar and M. Datcu, "Quantum annealer for network flow minimization in insar images," in *EUSAR 2021; 13th European Conference on Synthetic Aperture Radar*, 2021, pp. 1–4.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.