Improving Convergence for Quantum Variational Classifiers Using Weight Re-Mapping

Michael Kölle, Alessandro Giovagnoli, Jonas Stein, Maximilian Balthasar Mansky, Julian Hager and Claudia Linnhoff-Popien

Institute of Informatics, LMU Munich, Oettingenstraße 67, Munich, Germany

Keywords: Variational Quantum Circuits, Variational Classifier, Weight Re-Mapping.

Abstract: In recent years, quantum machine learning has seen a substantial increase in the use of variational quantum circuits (VQCs). VQCs are inspired by artificial neural networks, which achieve extraordinary performance in a wide range of AI tasks as massively parameterized function approximators. VQCs have already demonstrated promising results, for example, in generalization and the requirement for fewer parameters to train, by utilizing the more robust algorithmic toolbox available in quantum computing. A VQCs' trainable parameters or weights are usually used as angles in rotational gates and current gradient-based training methods do not account for that. We introduce weight re-mapping for VQCs, to unambiguously map the weights to an interval of length 2π, drawing inspiration from traditional ML, where data rescaling, or normalization techniques have demonstrated tremendous benefits in many circumstances. We employ a set of five functions and evaluate them on the Iris and Wine datasets using variational classifiers as an example. Our experiments show that weight re-mapping can improve convergence in all tested settings. Additionally, we were able to demonstrate that weight re-mapping increased test accuracy for the Wine dataset by 10% over using unmodified weights.

1 INTRODUCTION

Machine learning (ML) tasks are ubiquitous in a vast number of domains, including central challenges of humanity, such as drug discovery or climate change forecasting. In particular, ML techniques allow to tackle problems, for which computational solutions were unimaginable before its rise. While many tasks can be solved efficiently using machine learning techniques, fundamental limitations are apparent in others, e.g., the curse of dimensionality (Bellman, 1966). Inspired by the success of classical ML and a promising prospect to circumvent some of its intrinsic limitations, quantum machine learning (QML) has become a central field of research in the area of quantum computing. Based on the laws of quantum mechanics rather than classical mechanics, a richer algorithmic tool set can be used to solve some computational problems faster on a quantum computer than classically possible. In some special cases, this new technology allows for exponential speedups for specific problems, such as classification and regression, i.e., by employing a quantum algorithm to solve systems of linear equations (SLEs), whose runtime is logarithmic in the dimensions of the SLEs for sparse matrices (Harrow et al., 2009). Aside from the application of such provably efficient basic linear algebra subroutines, a central pillar of quantum computing, a new field in QML has recently been proposed: Variational Quantum Computing (VQC). Using elementary constructs of quantum computing, a universal function approximator can be constructed, much in the same way as in artificial neural networks. In analogy to logic gates, quantum gates are the fundamental algorithmic building blocks. The mathematical operations represented by quantum computing building blocks are different from classical elements, as they describe rotations and reflections, or concatenations of these in a vector space. Rather than working in a general vector space, quantum computing acts on a Hilbert space with exponentially more dimensions available with increasing number of parameters. In essence, every variational quantum circuit can be decomposed into a set of parametrized single qubit rotations and unparmeterized reflections (Nielsen and Chuang, 2010).

While variational quantum computing has benefits such as fewer training parameters to optimize (Du et al., 2020), many open problems remain in its practical application. The structure of a quantum circuit is its own problem, where many different concatena-

Kölle, M., Giovagnoli, A., Stein, J., Mansky, M., Hager, J. and Linnhoff-Popien, C.
Improving Convergence for Quantum Variational Classifiers Using Weight Re-Mapping.
DOI: 10.5220/0011696300003393
In *Proceedings of the 15th International Conference on Agents and Artificial Intelligence (ICAART 2023) - Volume 2*, pages 251-258
ISBN: 978-989-758-623-1; ISSN: 2184-433X
Copyright © 2023 by SCITEPRESS – Science and Technology Publications, Lda. Under CC license (CC BY-NC-ND 4.0)

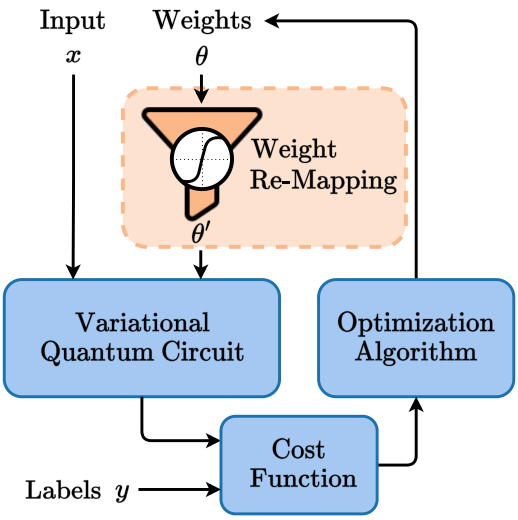


Figure 1: Overview of the Variational Quantum Circuit Training Process with Weight Constraints.

tions of quantum gates are possible and lead to different training results. At the same time, the Hilbert space is a closed space with possible values from $[0, 2\pi]$. It is still unclear how to best work with the classically unusual domain restriction of the parameters, set by the rotation operations. This problem is of great relevance, as rotations, being periodic functions, are not injective, and thus lead to ambiguous parameter assignments and thus worse training performance.

Drawing inspiration from classical ML, where data rescaling, or normalization techniques have shown immense improvements in many cases (Singh and Singh, 2019), we propose the introduction of re-mapping training parameters in variational quantum circuits as described in Figure 1. Concretely, we employ a set of well known fixed functions to unambiguously map the weights to an interval of length 2π and test their performance. For evaluation purposes, we use a classification problem and a circuit architecture suitable for the employed variational classifier. Our experimental data shows, that the proposed weight re-mapping leads to faster convergence in all tested settings compared to runs with unconstrained weights. In some cases, the overall test accuracy can also be improved. In this work, we first describe the basics of variational quantum circuits and related work. We then explain the idea behind our approach and how we setup up our experiments. Finally, we present and discuss our results and end with a summary, limitations and future work. All experiments and a PyTorch implementation of the used weight re-mapping functions can be found here[1].

[1]https://github.com/michaelkoelle/qw-map

2 VARIATIONAL QUANTUM CIRCUITS

The most prominent function approximator used in classical machine learning is the artificial neural network: a combination of parameterized linear transformations and typically non-linear activation functions, applied to neurons. The weights and biases used to parameterize the linear transformations can be updated using gradient based techniques like backpropagation, optimizing the approximation quality. According to Cybenko's universal approximation theorem, this model allows the approximation of arbitrary functions with arbitrary precision.(Cybenko, 1989).

In a quantum circuit, information is stored in the state of a qubit register $|\psi_i\rangle$, i.e., normalized vectors living in a Hilbert space $\mathcal{H}$. In quantum mechanics, a function mapping the initial state $|\psi_i\rangle$ onto the final state $|\psi_f\rangle$ is expressed by a unitary operator U that maps the inputs onto the outputs as in $|\psi_f\rangle = U|\psi_i\rangle$. In contrast to classical outputs, quantum outputs can only by obtained via so called measurements, which yield an eigenstate corresponding with an expected value of $\langle\psi_f| O |\psi_f\rangle$, where O is typically chosen to be the spin Hamiltonian in the z-axis.

In order to build a quantum function approximator in form of a VQC, one typically decomposes the arbitrary unitary operator U into a set of quantum gates. Analogously to Cybenko's theorem, in the quantum case it can be proved that any unitary operator acting on multiple qubits can always be expressed by the combination of controlled-not (CNOT) and rotational (ROT) gates, which represent reflections or rotations of the vector into the Hilbert space respectively (Nielsen and Chuang, 2010). While CNOTs are parameter-free gates, each rotation is characterized by the three angles around the axes of the Bloch sphere. These rotation parameters are the weights of the quantum variational circuit. We can thus say that the final state actually depends on the weights θ of the circuit, and rewrite the output final value as

$$\langle\psi_f(\theta)| O |\psi_f(\theta)\rangle \tag{1}$$

Starting from this theoretical basis, a function approximator can be obtained once a suitable circuit structure, also called ansatz, has been chosen. Once this is done and an objective function has been chosen, the rotation weights can be trained in a quantum-classical pipeline, as shown in Figure 1, completely analogously to what is done with a neural network.

Similar to a classical neural network, where the gradient is calculated using backpropagation, we can differentiate the circuit with respect to the parameters θ in a similar way using the parameter shift rule

(Schuld et al., 2019). It has to be noted however, that this approach has linear time complexity in the number of parameters and every gradient calculation has to be executed on a quantum computer. This is significantly worse than the constant time complexity regarding the number of parameters, needed in back-propagation.

3 RELATED WORK

The task of embedding data from $\mathbb{R}^n$ to $SU(2^n)$ has received limited attention so far, with approaches focusing on global embeddings (Lloyd et al., 2020) or local mappings. The former approach adds another classical calculation overhead to the whole VQC construction. It is also data specific and needs to be retrained for each dataset.

For local embeddings there are arguments for rescaling the data classically to a fixed interval as $[\min(\text{data}), \max(\text{data})] \rightarrow [0,1]$ for each data dimension and then mapping that to rotations on a single qubit axis (Stoudenmire and Schwab, 2016), multi-qubit embedding (Mitarai et al., 2018) and random linear maps (Wilson et al., 2019)(Benedetti et al., 2019).

4 OUR APPROACH

In this section, we first explain the idea and the process behind weight re-mapping for variational quantum circuits. Then we elaborate on the chosen example implementation of a variational quantum classifier architecture and the datasets on which we evaluate the approach. Note that, the approach is not limited to supervised learning tasks. Because weight re-mapping can be applied to VQCs that act as function approximators, it can be easily applied to other machine learning tasks such as unsupervised learning, reinforcement learning, and so on. Finally, we detail our training procedure and pre-tests that influenced our architecture decisions.

4.1 Weight Re-Mapping

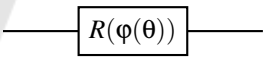
In a classical neural network the weights that characterize the connections between the neurons are represented by a vector $\vec{\theta} \in \mathbb{R}^n$. Once the neural network has been evaluated and the back-propagation has been performed, we obtain the gradient of the loss function with respect to the parameters $\vec{\theta}$, namely $\nabla_{\vec{\theta}} \mathcal{L}(\vec{\theta})$. The weights are then updated as follows:

$$\vec{\theta}_i = \vec{\theta}_{i-1} - \alpha \nabla_{\vec{\theta}_{i-1}} \mathcal{L}(\vec{\theta}_{i-1}) \tag{2}$$

This update step relies on the fact that the space in which we are moving to minimize the loss function is $\mathbb{R}^n$, and thus every value of the real number line can be assumed by the weights. The same is not true in the case of quantum variational circuits, where, as previously discussed, the parameters $\vec{\theta}$ represent rotations around one of the three axes in the space of the Bloch sphere. Since a rotation of angle θ around a generic direction $\hat{v}$ has a period of 2π, meaning $R(\hat{v}, \theta) = R(\vec{v}, \theta + 2\pi)$, it follows that for a parameter θ that is close enough to the boundaries of the period, updating the value according to (2) may result in making the parameter end up in an adjacent period and thus assume an unexpected value, possibly opposite to the desired direction. We can conclude that the natural space where the parameters of a quantum variational circuit should be taken is $[\phi, \phi + 2\pi]$, with $\phi \in \mathbb{R}$. To center the interval around the value 0 we pick $\phi = -\pi$. This way the parameters of a quantum variational circuit are $\vec{\theta} \in [-\pi, \pi]^n$. In order to constrain the parameters in the interval $[-\pi, \pi]$ we apply a *mapping function* to the weights. A mapping function is any function of the type

$$\varphi : \mathbb{R} \rightarrow [a, b] \tag{3}$$

that maps the real line to a compact interval. In our case we are interested in $a = -b = -\pi$, so that $\varphi : \mathbb{R} \rightarrow [-\pi, \pi]$. Introducing the mapping function means that, whenever a forward pass in performed, every rotation gate depending of a set of angles $\theta = (\theta_x, \theta_y, \theta_z)$ will first have its parameters remapped.

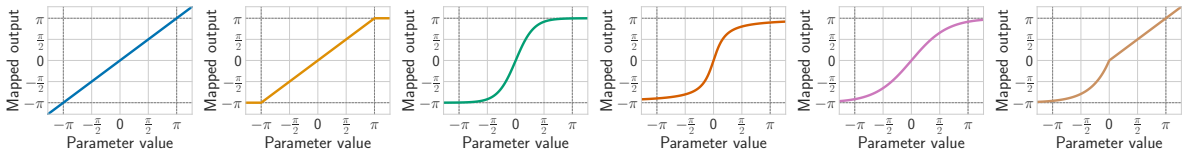


We want to emphasize that because the parameters $\vec{\theta}$ will be constrained in the forward pass, they are always free to take values from the real line. This way we can rewrite the basic update rule as follows:

$$\vec{\theta}_i = \vec{\theta}_{i-1} - \alpha \nabla_{\vec{\theta}_{i-1}} \mathcal{L}(\varphi(\vec{\theta}_{i-1})) \tag{4}$$

This shows that no mapping function is applied to the weights during the update step, while it is during the forward pass, represented by the loss function.

The mapping functions we decided to use in the experiments are some of the most common functions normally used to map the real numbers into a compact interval, properly scaled in such a way that the output interval is $[-\pi, \pi]$. First, we introduce the identity function, which serves as our baseline without a weight constraint. This is equal to applying no mapping at all, as can be seen in Figure 2a.



(a) No Re-Mapping. (b) Clamp. (c) Tanh. (d) Arctan. (e) Sigmoid. (f) Elu.

Figure 2: Re-Mapping functions.

$$\varphi_1(\theta) = \theta \tag{5}$$

Starting out with the first mapping function in Figure 2b, we simply clamp all parameter values above π to π and all values below $-\pi$ to $-\pi$. Within the interval $[-\pi, \pi]$ the identity function is used.

$$\varphi_2(\theta) = \begin{cases} -\pi & \text{if } \theta < -\pi \\ \pi & \text{if } \theta > \pi \\ \theta & \text{otherwise} \end{cases} \tag{6}$$

Next, we use the hyperbolic tangent function and apply a scaling of π. This creates a steeper mapping around $\theta = 0$ but has a smoother transition near $-\pi$ and π, as shown in Figure 2c

$$\varphi_3(\theta) = \pi \tanh(\theta) \tag{7}$$

After some initial tests, we wanted to explore the impact of how fast the function approaches the bounds $-\pi$ and π. Therefore, we scaled the inverse tangent function with factor 2 in both axes. Its graph is shown in Figure 2d.

$$\varphi_4(\theta) = 2 \arctan(2\theta) \tag{8}$$

Furthermore, we tested a modified sigmoid function, which is not as steep as the previous two and approaches the bounds faster than φ_2 but slower than φ_3. To archive the graph in Figure 2e, we needed to scale the function with factor 2π and shift it by $-\pi$.

$$\varphi_5(\theta) = \frac{2\pi}{1 + e^{-\theta}} - \pi \tag{9}$$

Lastly, we tested the Exponential Linear Unit or ELU as a asymmetric function, expecting that it should not work as good as the previously mentioned functions. To make sure that the function at least converges to the lower bound $-\pi$ for $\theta < 0$, we set its α parameter to π. Its graph can be seen in Figure 2f.

$$\varphi_6(\theta) = \begin{cases} \pi(e^{\theta} - 1) & \text{if } \theta < 0 \\ \theta & \text{otherwise} \end{cases} \tag{10}$$

4.2 Variational Circuit Architecture

The variational quantum circuit used in the variational classifier consists of 3 parts (Figure 3).

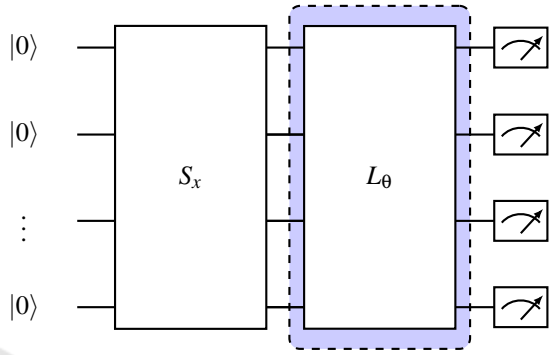
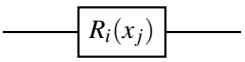


Figure 3: Abstract variational quantum circuit used in this work. Dashed blue area indicates repeated layers.

State Preparation

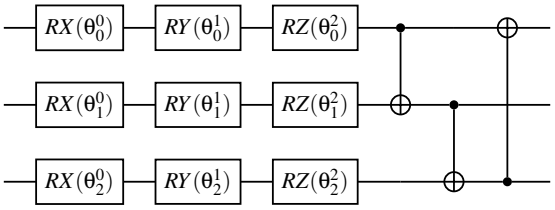
Initially, we start with the state preparation S_x, where the feature vector is embedded into the circuit. Here, real values of the feature vector are mapped from the Euclidean space to the Hilbert space. There are many ways to embed data into a circuit and many different state preparation methods. After a small scale pre-test, we determined that the Angle Embedding yielded the best results on both the Iris and Wine dataset. *Angle Embedding* is a simple technique to encode n real-valued features into n qubits. Each of the qubits are initialized as $|0\rangle$ and then rotated by some angle around either the x- or y-axis. The chosen angle corresponds to the feature amplitude to be encoded. Usually single axis rotational gates, are applied, as seen in the example



if we call $x_j \in \mathbb{R}$ the j-th feature to be embedded and R_i, with $i \in \{X, Y\}$ the rotational gate around the i-th axis, then the embedding is performed by simply applying the gate represented below. Note that the z-axis is left out: since the initial qubits are in the $|0\rangle$ state, which correspond in the Bloch sphere to vectors along the z-axis, rotations around it naturally have no effect and no information would be encoded.

Variational Layers

The second part of the circuit L_θ, hereafter called layers, consists of repeated single qubit rotations and entanglers (in Figure 3 everything contained within the dashed blue area is repeated L times, where L is the layer count). Specifically, we use a layer architecture inspired by the circuit-centric classifier design (Schuld et al., 2020). All circuits presented in this paper use three single qubit rotation gates and CNOT gates as entanglers. Here, we show an example of the first layer L_θ of a three qubit variational classifier.



In the circuit above θ_i^j denotes a trainable parameter where i represents the qubit index and $j \in \{0,1,2\}$ the index of the single qubit rotation gate. For clarity, we omitted the index l which denotes the current layer in the circuit. In each layer, the target bit of the CNOT gate is given by $(i+l) \bmod n$. For example, in the first layer $l = 1$ the control and target qubits are zero qubits apart and are direct neighbors in a circular manner. In the next layer $l = 2$, the control and target qubits are one qubit apart. Continuing the example, the index of the target qubit of qubit $i = 0$ is $(0+2) \bmod 3 = 2$.

Measurement

The last part of the circuit architecture is the measurement. We measure the expectation value in the computational basis (z) of the first k qubits where k is the number of classes to be determined. Then a bias is added to each measured expectation value. The biases are also part of the parameters of the VQC and are updated accordingly. Lastly, the softmax of the measurement vector is computed in order to normalize the probabilities.

4.3 Datasets

In this section, we present the datasets that were used to assess the variational classifier from section 4.2. We chose two popular datasets that pose a simple supervised learning classification task.

4.3.1 Iris Dataset

The Iris dataset originates from an article by Fisher (Fisher, 1936) in 1936. Since then it has been frequently referenced in many publications to this day. We are using a newer version of the dataset, where two small errors were fixed in comparison to the original article. The dataset includes 3 classes with 50 instances each, where each class refers to one of the following types of iris plants: Iris Setosa, Iris Versicolour, Iris Virginica. Each feature vector contains sepal length (in cm), sepal width (in cm), petal length (in cm) and petal width (in cm) respectively. The Iris dataset is a particularly easy dataset to classify, especially when considered that the first two classes are linearly separable. The latter two classes are not linearly separable.

Table 1: Hyperparameters used for each dataset.

Parameter	Iris Dataset	Wine Dataset
Learning Rate	0.0201	0.0300
Weight Decay	0.0372	0.0007
Batch Size	9	18
Embedding	Angle (X-Axis)	Angle (Y-Axis)
Number of Layers	8	9

4.3.2 Wine Dataset

The Wine dataset was created in July 1991 as a result of a chemical analysis of wines of three different cultivars grown in the same region in Italy (Vandeginste, 1990). Originally, they investigated 30 constituents but they were then lost to time and only a smaller version of the dataset still remains. The remaining dataset includes 3 classes which represent the different cultivars, a total of 178 instances containing 13 features each: Alcohol, Malic acid, Ash, Alcalinity of ash, Magnesium, Total phenols, Flavanoids, Nonflavanoid phenols, Proanthocyanins, Color intensity, Hue, OD280/OD315 of diluted wines and Proline. The dataset is not balanced and the instances are distributed among the classes as follows: class 1: 59, class 2: 71, class 3: 48.

4.4 Training

We started the training of the variational classifiers with a hyperparameter search using Bayesian optimization. We trained approximately 300 runs per dataset, optimizing the test accuracy. All runs in the hyperparamter search were executed without weight constraints. The hyperparameters of the best run for each dataset can be found in Table 1. We then ran 20 runs for every mapping function on the Iris dataset and 10 runs each on Wine dataset, each with the before mentioned hyperparameters. The initialization of the weights is important when using weight constraints, since the functions defined in Section 4.1 are centered around 0. Therefore, we initialized the

Table 2: Test Accuracy of tested mapping functions with 95% confidence interval.

Dataset	w/o Re-Mapping	Clamp	Tanh	Arctan	Sigmoid	ELU
Iris	0.953 ± 0.024	0.953 ± 0.024	0.953 ± 0.024	$\mathbf{0.957 \pm 0.023}$	0.950 ± 0.025	0.943 ± 0.026
Wine	0.617 ± 0.071	0.622 ± 0.071	0.706 ± 0.067	0.667 ± 0.069	$\mathbf{0.717 \pm 0.066}$	0.694 ± 0.067

Table 3: Accuracy (Validation) of tested mapping functions during convergence.

Iris Dataset						
Samples	w/o Re-Mapping	Clamp	Tanh	Arctan	Sigmoid	ELU
120	0.693	0.700	0.903	**0.913**	0.837	0.840
240	0.883	0.887	**0.937**	0.927	0.927	0.910
480	0.933	**0.937**	0.927	0.933	0.930	0.920
Wine Dataset						
Samples	w/o Re-Mapping	Clamp	Tanh	Arctan	Sigmoid	ELU
568	0.372	0.372	**0.533**	0.522	0.489	0.511
994	0.511	0.517	**0.639**	0.539	0.583	0.533
1846	0.572	0.572	**0.656**	0.639	0.628	0.628

weights close to 0 in the range of $[-0.01, 0.01]$. For the training we used Cross Entropy Loss and Adam Optimizer.

5 EXPERIMENTS

In this section we present our conducted experiments on the Iris and Wine datasets. To show the impact of the re-mapping functions from Section 4.1, we chose simple supervised learning tasks using a quantum variational classifier. However, in principle our approach can be used for any scenario that relies on a variational quantum circuit, for example Quantum Reinforcement Learning, QAOA, and more. We trained the classifier on two different datasets Iris (Section 4.3.1) and Wine (Section 4.3.2). All training related details can be found in Section 4.4. We first investigated the convergence behavior with and without the weight re-mapping. This is an important topic, especially for quantum machine learning, since the use of quantum hardware is much more expensive than classical hardware at the time of writing. In Figure 4 and Table 3 we show that faster convergence can be achieved by using weight re-mapping in all tested settings. With a faster convergence we can potentially save on compute which results in less power consumption and $C0_2$ emissions. We also investigated overall test accuracy in Table 2, where we can see minor improvements for the Iris dataset and a 10% improvement for the Wine dataset, compared to using no weight re-mapping.

5.1 Iris Dataset

In the following, we present the results of our experiments on the Iris dataset. In a pre-test we have found that Angle Embedding with rotations around the x-axis works best for the classification of the Iris dataset, but also requires more qubits than other embedding methods like Amplitude Embedding (Möttönen et al., 2005). It should be noted, that Angle Embedding with rotations around the y-axis should, in theory, perform similarly. Angle Embedding with rotations around the z-axis does not work at all, since the rotations are invariant to the expectation value of the measurement w.r.t. z-axis. We then ran a quick hyperparameter search on the architecture without a re-mapping function, determining the hyperparameters listed in Table 1. We wanted to see if we can further improve the convergence of an architecture with an already good set of hyperparameters, just by introducing weight re-mapping. We trained the model with and without weight re-mapping with 20 different seeds on the dataset. The whole training curves can be found in Figure 4 and the convergence behavior can be seen in detail in Table 3, where we picked three points during convergence and compared the accuracies. We observe a increase of over 20% using the Arctan re-mapping compared to using no re-mapping after 120 samples. Similarly, after 240 samples the model Tanh mapping has already converged and resulted in over 5% higher accuracy. As the model without re-mapping reaches convergence after 480 samples, we see practically no difference between the approaches. Theses differences can be attributable to the re-mapping functions. In this particular case the optimal weights are close to zero and especially Tanh and Arctan are very steep in this area. One possibility is that the re-mapping kept the values closer to this area resulting in finding the weights faster. We also investigated if the re-mapping process has any impact on the overall test accuracy, where the results can be found in Table 2. As expected, the test accuracy did not decrease when using weight re-mapping for the Iris dataset. It even reached a minimally higher test accuracy which is due to more time for fine-tuning after the earlier convergence. In this setting, Arctan is our overall preferred choice because of its fast convergence and second best test accuracy.

5.2 Wine Dataset

The results of our experiments on the Wine dataset are presented below. In a preliminary test, we discovered that, in contrast to the Iris dataset, Angle Embedding

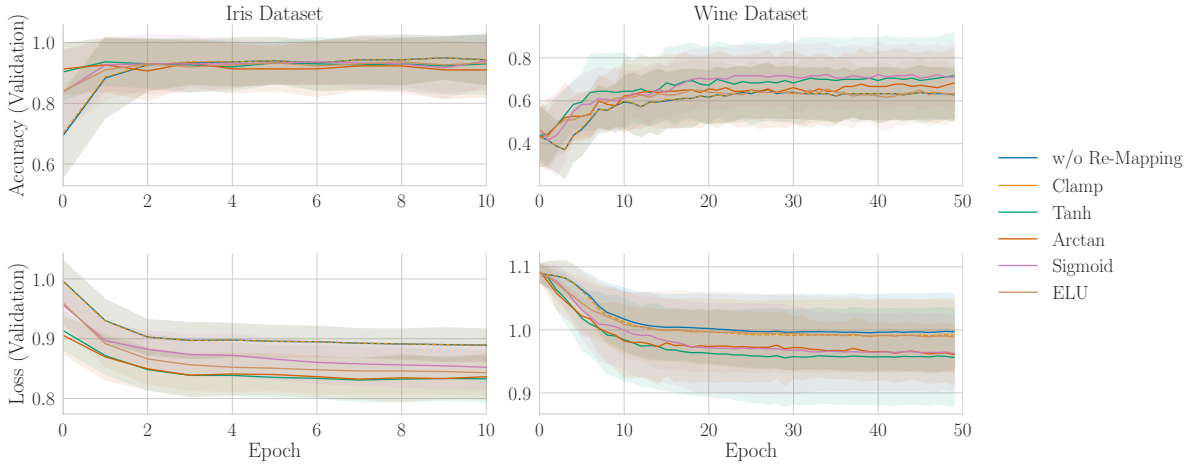


Figure 4: Validation curves for datasets Iris and Wine. In each epoch the algorithm processes 80% of the total samples of each dataset for training. This accounts to 120 samples for the Iris dataset and 142 samples for the Wine dataset.

with rotations around the y-axis works best. In theory, Angle Embedding with rotations around the x-axis should perform similarly, since we are measuring the expectation value w.r.t. z-axis and rotations around the z-axis do not work at all, since the rotations are invariant to the expectation value of the measurement. Like the architecture used for the Iris dataset, Angle Embedding outperformed more efficient embedding methods like Amplitude Embedding (Möttönen et al., 2005). The architecture without a re-mapping function was then also subjected to a small scale hyperparameter search, yielding the hyperparameters listed in Table 1. We were interested in determining whether adding weight re-mapping may further enhance the convergence of an architecture with a good set of hyperparameters. On the dataset, we trained the model both with and without weight re-mapping using 20 seeds. The entire training curves are shown in Figure 4, and Table 3 shows the convergence behavior in detail, comparing the accuracies at three points during convergence. From the results, we can see that Tanh is outperforming all other tested settings during convergence and models with weights re-mapping generally converging faster than the baseline. At 568 samples Tanh has a 15% higher accuracy compared to the model without weight re-mapping. After 994 samples Tanh is almost fully converged, while the baseline only converges long after 1846 samples. We believe that the cause of this effect is the same as that mentioned in the section above. These differences are brought on by the re-mapping operation. In this case, the ideal weights are almost zero, and Tanh is very steep in this region, mapping the values close to zero and enabling a more detailed search in this region, ultimately resulting in a faster convergence. Additionally, Table 2 shows that the weight re-mapping not only leads to a faster convergence but also, in this case, a 10% improvement in test accuracy, closely followed by Tanh. Tanh is our first preference in this situation because of its quick convergence and second-best test accuracy.

6 CONCLUSION

In this work, we introduced weight re-mapping for variational quantum circuits, a novel method for accelerating convergence using quantum variational classifiers as an example. In order to test our classifier on two datasets — Iris and Wine — we first conducted an initial hyperparameter search to identify a good set of parameters without re-mapping. After that, we introduced our weight re-mapping approach for evaluation. Our experiments show that weight re-mapping can improve convergence in all tested settings. Faster convergence may allow us to lower the amount of compute we need, which will reduce power consumption and $C0_2$ emissions. This is crucial, especially for quantum machine learning, where access to quantum hardware is quite expensive. Furthermore, for the Wine dataset, we were able to demonstrate that using weight re-mapping improved test accuracy by 10% compared to using unmodified weights. For both settings, we would recommend a re-mapping function that is steep around the initialization point, such as Arctan and Tanh.

There are additional opportunities based on our findings. This includes testing other datasets, with more features or more complex classification problems. We also see opportunities to develop specialized optimizers that take into account the unique char-

acteristics of parameters in quantum circuits. Finally, more research is required to determine the impact of weight re-mapping on other tasks and fields, such as Quantum Reinforcement Learning.

REFERENCES

Bellman, R. (1966). Dynamic programming. *Science*, 153(3731):34–37.

Benedetti, M., Lloyd, E., Sack, S., and Fiorentini, M. (2019). Parameterized quantum circuits as machine learning models. *Quantum Science and Technology*, 4(4):043001.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Math. Control. Signals Syst.*, 2(4):303–314.

Du, Y., Hsieh, M.-H., Liu, T., and Tao, D. (2020). Expressive power of parametrized quantum circuits. *Physical Review Research*, 2(3).

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

Harrow, A. W., Hassidim, A., and Lloyd, S. (2009). Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15).

Lloyd, S., Schuld, M., Ijaz, A., Izaac, J., and Killoran, N. (2020). Quantum embeddings for machine learning. arXiv:2001.03622 [quant-ph].

Mitarai, K., Negoro, M., Kitagawa, M., and Fujii, K. (2018). Quantum circuit learning. *Phys. Rev. A*, 98:032309.

Möttönen, M., Vartiainen, J. J., Bergholm, V., and Salomaa, M. M. (2005). Transformation of quantum states using uniformly controlled rotations. *Quantum Info. Comput.*, 5(6):467–473.

Nielsen, M. A. and Chuang, I. L. (2010). *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press.

Schuld, M., Bergholm, V., Gogolin, C., Izaac, J., and Killoran, N. (2019). Evaluating analytic gradients on quantum hardware. *Physical Review A*, 99(3):032331. arXiv:1811.11184 [quant-ph].

Schuld, M., Bocharov, A., Svore, K. M., and Wiebe, N. (2020). Circuit-centric quantum classifiers. *Physical Review A*, 101(3).

Singh, D. and Singh, B. (2019). Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, page 105524.

Stoudenmire, E. and Schwab, D. J. (2016). Supervised Learning with Tensor Networks. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Vandeginste, B. (1990). Parvus: An extendable package of programs for data exploration, classification and correlation, m. forina, r. leardi, c. armanino and s. lanteri, elsevier, amsterdam, 1988, price: Us $645 isbn 0-444-43012-1. *Journal of Chemometrics*, 4(2):191–193.

Wilson, C. M., Otterbach, J. S., Tezak, N., Smith, R. S., Polloreno, A. M., Karalekas, P. J., Heidel, S., Alam, M. S., Crooks, G. E., and da Silva, M. P. (2019). Quantum Kitchen Sinks: An algorithm for machine learning on near-term quantum computers. arXiv:1806.08321 [quant-ph].

SCITEPRESS SCIENCE AND TECHNOLOGY PUBLICATIONS