Outline

This project consists of two parts. Part 1 contains 5 mandatory questions, each of which is marked out of 2, and it comprises 20% of the final mark for the project. For part 2 you choose one out of 5 topics, it comprises 80% of the final mark for the project, and is marked out of 10.

At the end of this project **you will have to turn in a report**. The report should contain your solutions to the 5 mandatory questions and a write up on the topic of your choice in part 2. This write up should not be longer than 10 pages, but rather it should be concise and to the point. Also, please do not forget to submit you code for part 2, as this will be taken into account when grading. **The tentative deadline for this project is May 22 (it may be extended).**

Part 1

1. State the *primal* and *dual problem* of the optimization that appears when training a support-vector machine with feature map ϕ and the corresponding kernel.

2. Give the definition of a *positive semidefinite kernel* as it appears in the context of support-vector machines.

3. Consider two unitaries U_1 and U_2 that prepare the n-qubit quantum states $|\psi\rangle$ and $|\phi\rangle$ from the state $|0^n\rangle$, respectively, i.e.,

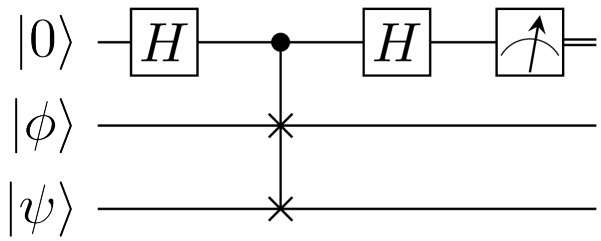
$$U_1 |0^n\rangle = |\psi\rangle ,$$

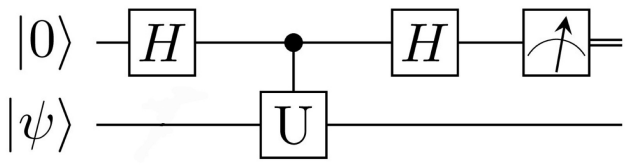
$$U_2 |0^n\rangle = |\phi\rangle .$$

One can see that estimating the probability of observing (i.e., 'measuring') the state $|0^n\rangle$, from the state $U_2^\dagger U_1 |0\rangle$ can be used to estimate the overlap (i.e., inner product) $|\langle\psi|\phi\rangle|$.

Remark. *Since global phase cannot influence quantum measurement outcomes, to discuss the phase/sign of the mathematical inner product as well, extra assumptions and tricks are needed.*

There are other ways of doing this, each requiring different resources. Namely, to estimate the overlap (i.e., inner product) between two quantum states $|\psi\rangle$ and $|\phi\rangle$, consider the following circuits





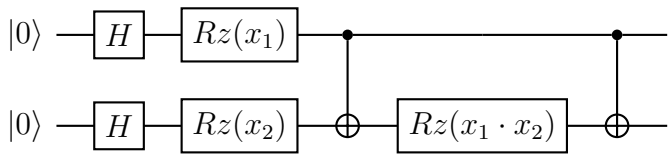
where the unitary U satisfies $U|\psi\rangle = |\phi\rangle$. These are called the SWAP and the Hadamard test, respectively. In the SWAP test, we use the controlled-SWAP gate.

Give the measurement outcome probabilities of both circuits when measuring the first qubit in the computational basis, and explain how they can be used to estimate (some aspect) of the overlap (i.e., inner product) of $|\psi\rangle$ and $|\phi\rangle$.

Hints.

- *https://www.cs.umd.edu/~amchilds/talks/qalg.pdf, pg. 31.*
- *https://arxiv.org/pdf/1804.03719.pdf, pg. 36.*

4. Compute the feature vector (i.e., the explicit 4-dimensional state vector) of a datapoint given by $\vec{x} = (x_1, x_2)$ for the following feature map circuit:



where $Rz(\phi)$ denotes a Z-rotation with angle ϕ

5. (Hard question, extra credit) Consider an n-qubit circuit of depth d (not depending on n). Show that you can compute the probability of the first qubit being in the state $|0\rangle$, *in time which does not depend on of* n, using only a classical computer.

Part 2

Supervised learning projects

Project 1. (*Preprocessing and benchmarking with real-world datasets*).

In this project you will compare ways of preprocessing high dimensional real-world datasets. Examples of such datasets are the Wine[1], BreastCancer[2] or MNIST[3] dataset. During preprocessing, you will have to reduce the dimension of these datasets in order to fit them on a limited number of qubits (say ≤ 9). Finally, you will benchmarking the performance of your techniques using cross-validation.

Objectives and tasks:

- Understand how principal component analysis (PCA)[4] can be used to reduce the dimension of a dataset and write code that does so for Wine[1], BreastCancer[2] and/or MNIST[3].
- Write code that benchmarks the resulting models (i.e., PCA + parameterized quantum circuits) using cross-validation (5-fold is enough) in order to measure the training and testing accuracies.
- (H) Design and employ other techniques, such as a trained neural network, to reduce the dimension of these datasets and compare their performance to that of PCA.

The report should contain:

1. An explanation of how PCA is used to reduce the dimension of datasets.
2. Code that uses PCA to reduce the dimension of high dimensional real-world datasets such as Wine[1], BreastCancer[2] or MNIST[3] in order to fit it on a limited number of qubits.

   Remark. *You are allowed to use the functions available in sci-kit learn, i.e,* ***https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html*** .
3. Benchmarks of the performance of your model (i.e., PCA + parameterized quantum circuit) using cross validation to measure the training and testing accuracies.

For a 8+: Design and employ another technique to reduce the dimension of these datasets and compare its performance to PCA.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[1] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html

[2] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html

[3] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html#sklearn.datasets.load_digits

Project 2. (*Hyperparameter testing in the explicit model*).

In this project you will compare how different hyperparameters affect the performance of the explicit model by comparing the performance of different configurations of these hyperparameters using cross validation (5 fold is enough) on standard datasets such as Circles[4], Moons[5] and Blobs[6].

Literature:

- `https://arxiv.org/pdf/1804.11326.pdf`
- `https://arxiv.org/pdf/1804.00633.pdf`

Objectives and tasks: Experiment how the following hyperparameters (or ones that you come up with yourself) affect the performance of the explicit model by using cross validation on standard datasets such as Circles[4], Moons[5] and Blobs[6] to measure training/testing-accuracies.

- *Encoding circuit:* While keeping the hyperparameters of the variational part fixed:
  - compare different 'preprocessing of the input', i.e., different maps Φ such that

    datapoint $x \in \mathbb{R}^n \mapsto$ rotation parameter $\Phi(x) \in [0, 2\pi)^n \mapsto$ encoding circuit $\mathcal{U}(\Phi(x))$.

    Remark. *Note that here it is not about reducing the dimension of x, just about prescaling/preprocessing raw-input x into valid rotation-angles $\Phi(x)$.*
  - compare different 'depths' of the encoding circuit, i.e., repetitions of $U(\Phi(x))$ in

    $$\mathcal{U}(\Phi(x)) = H^{\otimes n}U(\Phi(x))H^{\otimes n}U(\Phi(x))H^{\otimes n}\dots U(\Phi(x))H^{\otimes n}.$$
  - compare the effects of multi-qubit interactions (i.e., two-qubit entangling gates) in the encoding part of the circuit. That is, compare the difference between having multi-qubit interactions and not having them.
- *Variational circuit:* While keeping the hyperparameters of the encoding part fixed:
  - compare different 'depths' of the variational circuit, i.e., repetitions of $U^{(i)}_{loc}(\theta)$ in

    $$W(\theta) = U^{(1)}_{loc}(\theta)U_{ent}U^{(2)}_{loc}(\theta)U_{ent}U^{(3)}_{loc}(\theta)\dots U_{ent}U^{(\ell)}_{loc}(\theta),$$

    where U_{ent} denotes a layer of entangling gates.
  - compare different connectivities of the entangling layer, i.e., connectivities E in the entangling layer given by:

    $$U_{ent} = \bigotimes_{(i,j)\in E} CZ(i,j),$$

    where $CZ(i,j)$ denotes the controlled-Z gate on the i-th and j-th qubit.
- (H) *Data re-uploading:* Study the effect of using data re-uploading[7] on the performance of your model (while keeping all other the hyperparameters fixed).

The report should contain:

1. An explanation of the explicit models and its hyperparameters.
2. Code that implements the explicit model and that allows one to experiment with the relevant hyperparameters.
3. Benchmarks of the performance of the explicit model with different hyperparameter settings using cross validation to measure the training and testing accuracies.

For an 8.5+: Implement some aspect of data re-uploading[7] and study the effect on the performance.

[4] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_circles.html#sklearn.datasets.make_circles

[5] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html#sklearn.datasets.make_moons

[6] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html#sklearn.datasets.make_blobs

[7] https://arxiv.org/pdf/1907.02085.pdf

Project 3. (*Comparison between the implicit and the explicit model*).

In this project you will compare the performance of the implicit model (i.e., the model corresponding to the dual problem in SVMs) with the explicit model (i.e., the model corresponding to the primal problem in SVMs) using cross validation.

Literature:

- `https://arxiv.org/pdf/1804.11326.pdf`

Objectives and tasks:

- Study and implement the implicit model, i.e., the SVM classifier that arises from the dual problem using the kernel obtained from a parameterized quantum circuit.
- Write code that benchmarks both the implicit and explicit model using cross-validation to measure the training and testing accuracies.
- Give a fair comparisson between the performance of the explicit and implicit model. (You can use the datasets specified for the other projects).

The report should contain:

1. An explanation of the implicit model.
2. Code that implements the implicit model.
3. A comparison of the performance of the explicit and implicit model using cross validation techniques to measure the training and testing accuracies.

For an 8+: Write the report as stated above.

Generative modelling projects

Project 4. (*Sparse and deep versus shallow QCBMs*).

In this project you will compare the performances of QCBMs that are deep but sparse (i.e., only a few of its parameters are nonzero) with ones that are shallow.

Literature:

- `https://arxiv.org/pdf/1804.04168.pdf`

Objectives and tasks:

- Write code that implements and trains QCBMs.
- Generate a shallow QCBM (i.e., one with few parameters) and initialize it with randomly chosen parameters (i.e., do **not** train it).
- Generate and train a second deeper QCBM (i.e., one with more parameters) on samples generated by the previous QCBM and benchmark its performance.
- Try to enforce sparsity for the deeper QCBM. That is, train the deeper QCBM while making sure that only few of its parameters are nonzero (e.g., by altering the cost function or setting parameters that are small to zero).
- Benchmark and test whether the deep but sparse QCBM still performs well.

The report should contain:

1. An explanation and implementation of QCBMs.
2. Experimental results on training a deeper QCBM to learn the distribution of a randomly initialized shallow QCBM.
3. A discussion of why sparsity (i.e., regularization) is important for ML.
4. Experimental results on whether the deeper QCBM can 'learn' a shallow QCBM when only being allowed to use sparse parameters (i.e., few nonzero parameters).

For an 8+ and possibly a paper: Include point 4. into your report.

Project 5. (*Quantum generative adversarial network*).

In this project you will extend the QCBM implementation of the tutorial to a an implementation of a Quantum Generative Adverserial Network (QGAN).

Literature:

- `https://en.wikipedia.org/wiki/Generative_adversarial_network`
- `https://arxiv.org/abs/1904.00043`

Objectives and tasks:

- Learn what a generative adverserial network is, i.e., how it works and how to train it.
- Extend the QCBM implementation of the tutorial into a QGAN (i.e., the QCBM takes the role of the "generator").

  Remark. *You are allowed to use the implicit model classifier of the tutorial in order to implement the "discriminator".*

- Perform benchmarking experiments of your QGAN.

The report should contain:

1. An explanation of GANs and QGANs.
2. An implementation of a QGAN that extends the QCBM of the tutorial (i.e., a QGAN that uses the QCBM as the "generator").
3. Experimental results of the performance of the QGAN.

For a 8+ and possibly a paper: write the report as stated above.