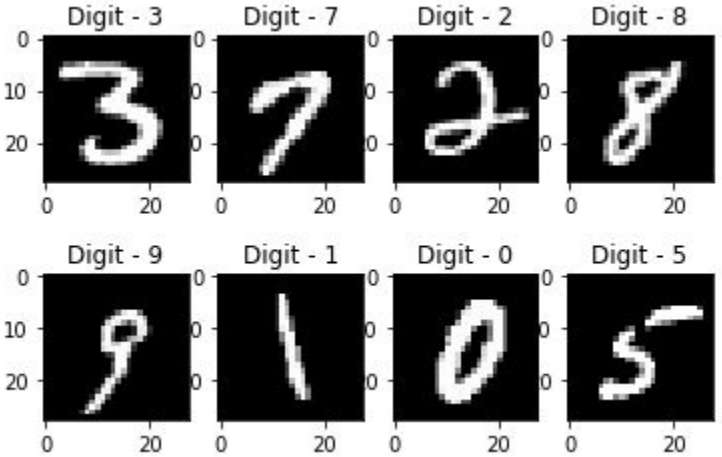
Quantum ensemble

Erinn and Andrew

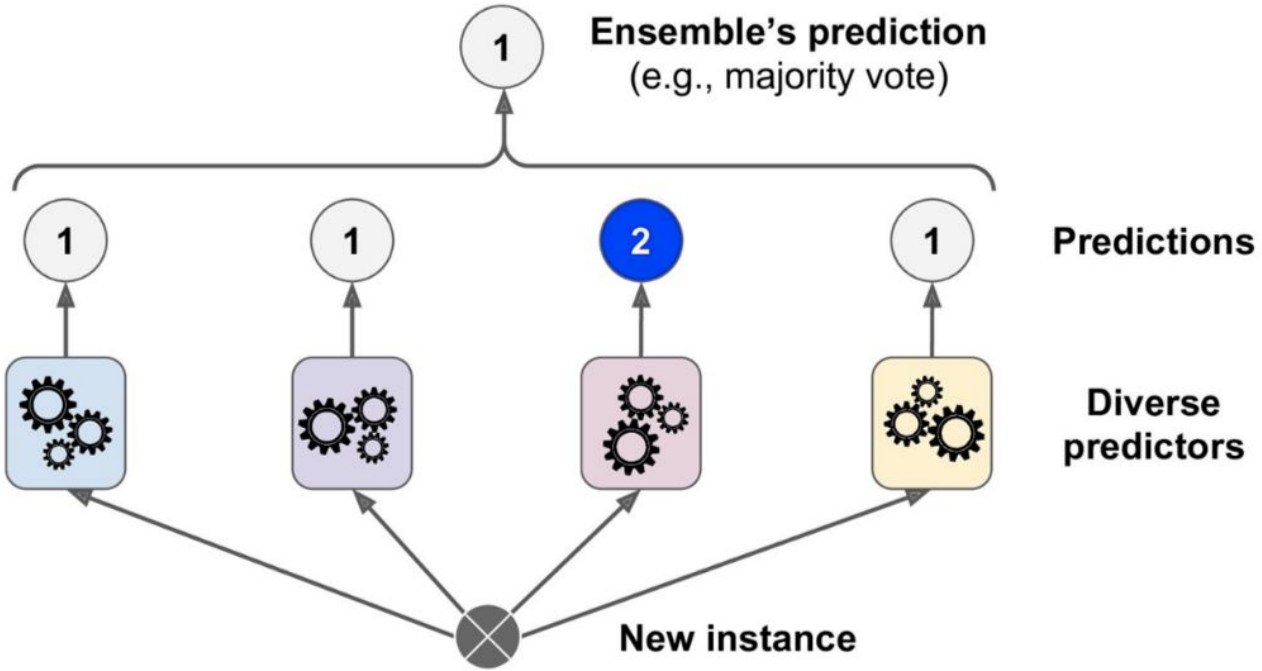
Classification problem in machine learning

- Idea: Given N class-labelled training data, want to train a model function/classifier *f* to predict which class a test data belongs to.
- Examples: Image classification



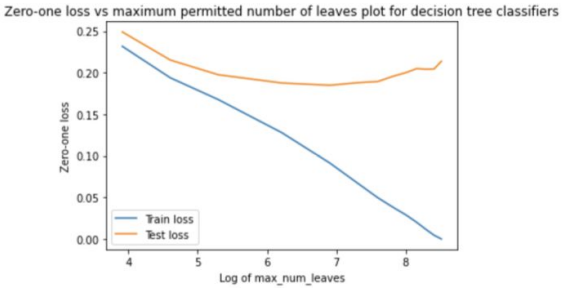
Ensemble methods for machine learning

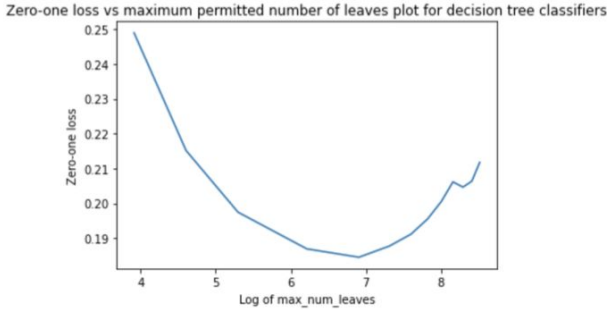
- Ensemble methods = Collection of classifiers

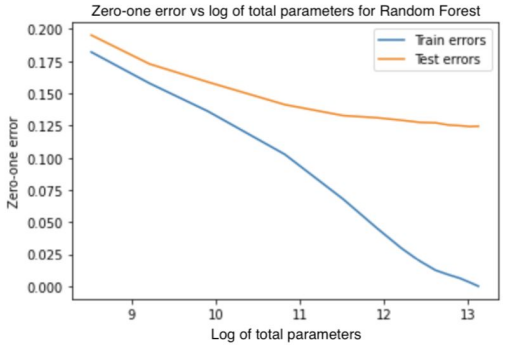


Why ensemble methods for machine learning?

- **Mean squared error = variance + bias**
- **Variance ~ overfitting**
- **Tackle the problem of overfitting - highly flexible models risk fitting to noise rather than capturing underlying structure.**
- Consider decision tree vs random forest:



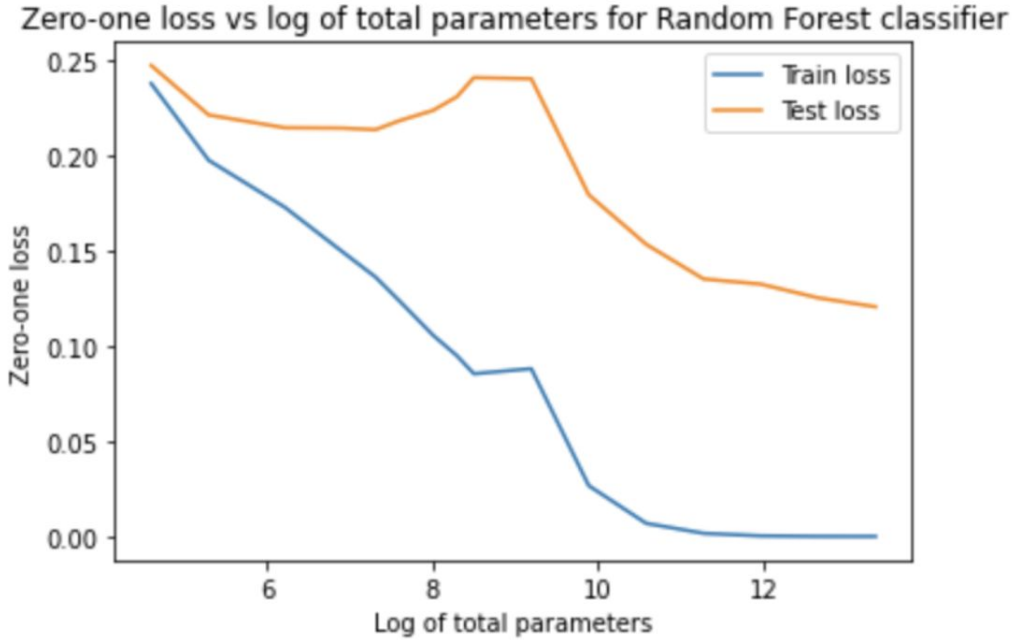




Individual decision tree classifiers are prone to overfitting and require pruning. Random forest leverage statistical averaging to decrease variance of prediction while maintaining low bias.

Classical double descent

- Increase parameters of single decision tree until overfits, then multiply decision trees

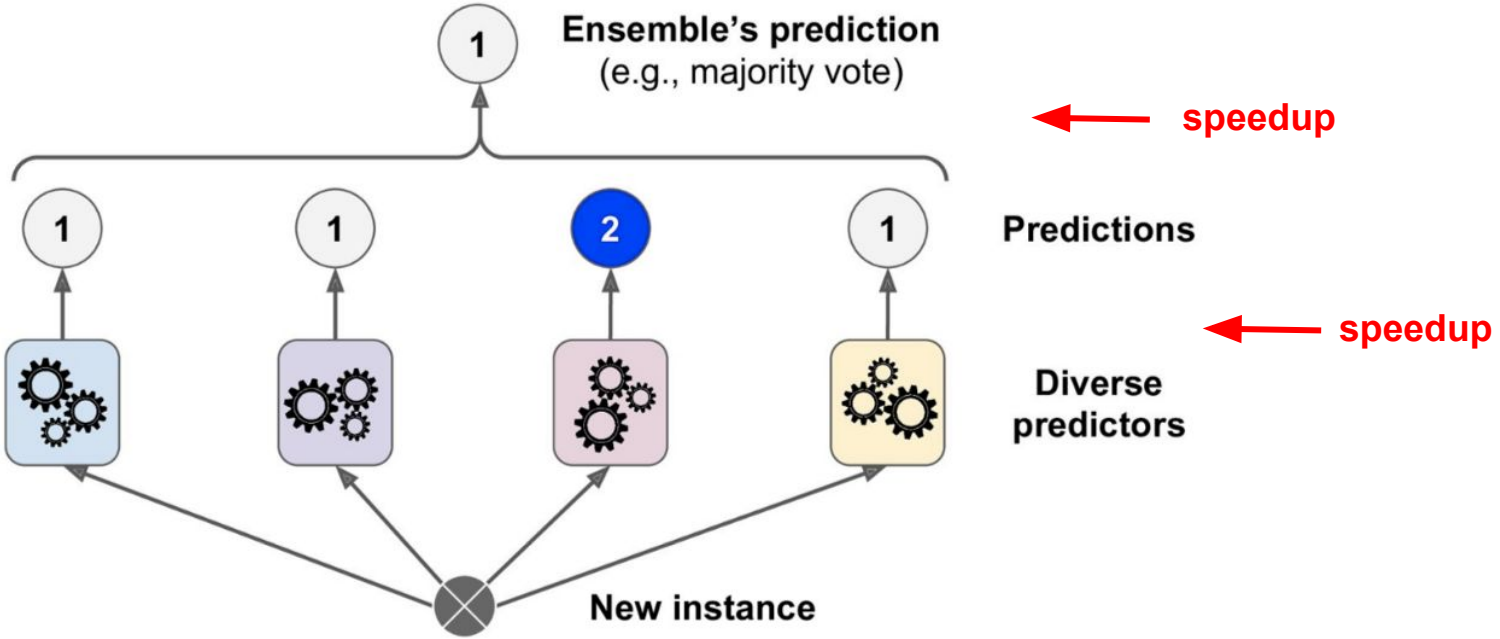


Sufficient and necessary condition for ensemble to outperform individual classifiers

- **Sub-classifiers must be accurate (>50% accuracy) and diverse (decorrelated).**

Why quantum ensemble?

- Exponential speedup in constructing individual classifiers + performing predictions + aggregating results



Overview of the quantum ensemble algorithm

1. **State preparation** → Encodes feature vectors into qubits
2. **Sampling in superposition** → In d steps, generates 2^d different trajectories, with each trajectory potentially corresponding to a different model.
3. **Learning via interference** → Uses a classifier F (e.g. cosine similarity) and propagates similarity calculation simultaneously to all 2^d trajectories.
4. **Measurement** → Single execution of measurement collapses all trajectories into a final prediction (0 or 1).

Quantum ensemble circuitry

1. State preparation

- Start with d control registers, 2*N data registers (N for training features + N for training labels).
- Use Hadamard gates to transform every qubit in control register to uniform superposition of |0> and |1>.
- Encode training set (x,y) into data register using some mapping S(x,y)|0> = |x,y>.

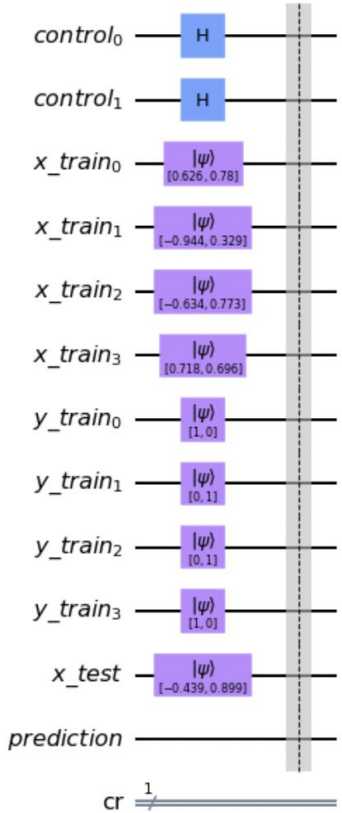
$$|\Phi_0\rangle = \left(W \otimes S_{(x,y)}\right) \bigotimes_{i=1}^{d} |0\rangle \otimes |0\rangle = \left(H^{\otimes d} \otimes S_{(x,y)}\right) \bigotimes_{i=1}^{d} |0\rangle \otimes |0\rangle = \bigotimes_{i=1}^{d} |c_i\rangle \otimes |x,y\rangle$$

$$|c_i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |+\rangle$$

- For now, we use the trivial encoding scheme for S(x,y):

$$S_{x_i}|0\rangle = \frac{x_{i1}}{\sqrt{x_{i1}^2 + x_{i2}^2}}|0\rangle + \frac{x_{i2}}{\sqrt{x_{i1}^2 + x_{i2}^2}}|1\rangle \equiv |x_i^{train}\rangle \qquad S_{y_i}|0\rangle = |y_i^{train}\rangle$$

1. State preparation: Circuitry



2. Sampling in superposition

- Entangles the data path with each of the d control qubits (uniform superpositions of |0> and |1>), generating 2^d different feature transformations to the data path (mathematically, you get 2^d different terms in superposition after this stage), each corresponding to a different model.

$$
\begin{aligned}
|\Phi_{i,1}\rangle &= \left(CU_{(i,1)}\right)|c_i\rangle \otimes |x,y\rangle \\
&= \left(CU_{(i,1)}\right)\frac{1}{\sqrt{2}}\left(|0\rangle + |1\rangle\right) \otimes |x,y\rangle \\
&= \frac{1}{\sqrt{2}}\left(|0\rangle |x,y\rangle + |1\rangle U_{(i,1)}|x,y\rangle\right)
\end{aligned}
$$

$$
\begin{aligned}
|\Phi_{i,2}\rangle &= (X \otimes \mathbb{1})|\Phi_{i,1}\rangle \\
&= \frac{1}{\sqrt{2}}\left(|1\rangle |x,y\rangle + |0\rangle U_{(i,1)}|x,y\rangle\right)
\end{aligned}
$$

$$
\begin{aligned}
|\Phi_{i}\rangle &= \left(CU_{(i,2)}\right)|\Phi_{i,2}\rangle \\
&= \left(CU_{(i,2)}\right)\frac{1}{\sqrt{2}}\left(|1\rangle |x,y\rangle + |0\rangle U_{(i,1)}|x,y\rangle\right) \\
&= \frac{1}{\sqrt{2}}\left(|1\rangle U_{(i,2)}|x,y\rangle + |0\rangle U_{(i,1)}|x,y\rangle\right).
\end{aligned}
$$

$$
|\Phi_d\rangle = \frac{1}{\sqrt{2^d}}\sum_{b=1}^{2^d}|b\rangle V_b|x,y\rangle = \frac{1}{\sqrt{2^d}}\sum_{b=1}^{2^d}|b\rangle |x_b,y_b\rangle
$$

In general, the unitary operators U(i,j) can be randomized and perform feature transformations to map input data into latent embeddings.

2. Sampling in superposition

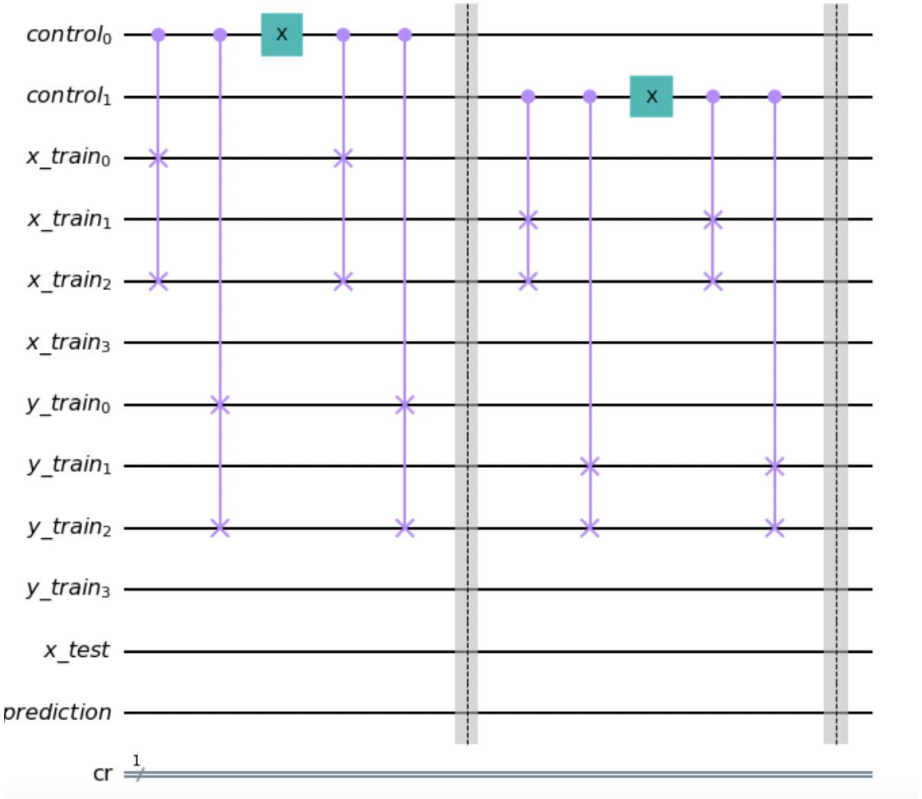
- An example sampling scheme for 4 training data. This ensures that each trajectory is independent from each other.

$$
\begin{aligned}
U_{(1,1)} &= \mathrm{SWAP}(x_0, x_2) \times \mathrm{SWAP}(y_0, y_2); \\
U_{(1,2)} &= \mathrm{SWAP}(x_1, x_3) \times \mathrm{SWAP}(y_1, y_3); \\
U_{(2,1)} &= \mathbb{1}; \\
U_{(2,2)} &= \mathrm{SWAP}(x_2, x_3) \times \mathrm{SWAP}(y_2, y_3);
\end{aligned}
$$

$$
\begin{aligned}
|\Phi_1\rangle = \frac{1}{2}\Big[& |00\rangle\, U_{(2,1)}U_{(1,1)}\, |x_0, x_1, x_2, x_3\rangle\, |y_0, y_1, y_2, y_3\rangle \\
& + |01\rangle\, U_{(2,1)}U_{(1,2)}\, |x_0, x_1, x_2, x_3\rangle\, |y_0, y_1, y_2, y_3\rangle \\
& + |10\rangle\, U_{(2,2)}U_{(1,1)}\, |x_0, x_1, x_2, x_3\rangle\, |y_0, y_1, y_2, y_3\rangle \\
& + |11\rangle\, U_{(2,2)}U_{(1,2)}\, |x_0, x_1, x_2, x_3\rangle\, |y_0, y_1, y_2, y_3\rangle \Big].
\end{aligned}
\quad \longrightarrow \quad
\begin{aligned}
|\Phi_2\rangle = \frac{1}{2}\Big[& |11\rangle\, |x_0, x_3, x_1, x_2\rangle\, |y_0, y_3, y_1, y_2\rangle \\
& + |10\rangle\, |x_2, x_1, x_3, x_0\rangle\, |y_2, y_1, y_3, y_0\rangle \\
& + |01\rangle\, |x_0, x_3, x_2, x_1\rangle\, |y_0, y_3, y_2, y_1\rangle \\
& + |00\rangle\, |x_2, x_1, x_0, x_3\rangle\, |y_2, y_1, y_0, y_3\rangle \Big].
\end{aligned}
$$

The quantum cosine classifier then selects a random entry of this superposition. For example, choosing the third entry yields x1, x3, x2, x0.

2. Sampling in superposition



3. Quantum Cosine Classifier

Classical Cosine Classifier

- Define d(x, y) = <x, y> / (||x||×||y||), i.e. the cosine distance between x and y.
- Given a training data point, (x_b, y_b) and a testing data point, $x^{(test)}$, evaluate the probability of $y^{(test)} = y_b$:

$$Pr\left(y^{(\text{test})} = y_b\right) = \frac{1}{2} + \frac{\left[d\left(x_b, x^{(\text{test})}\right)\right]^2}{2}$$

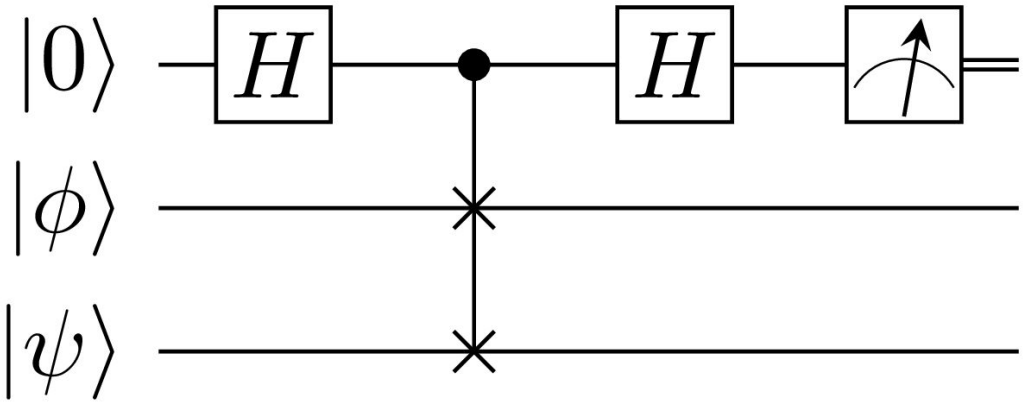
- The final classification rule:

$$y^{(\text{test})} = \begin{cases} y_b, & \text{if } Pr\left(y^{(\text{test})} = y_b\right) > \frac{1}{2} \\ 1 - y_b, & \text{otherwise} \end{cases}$$

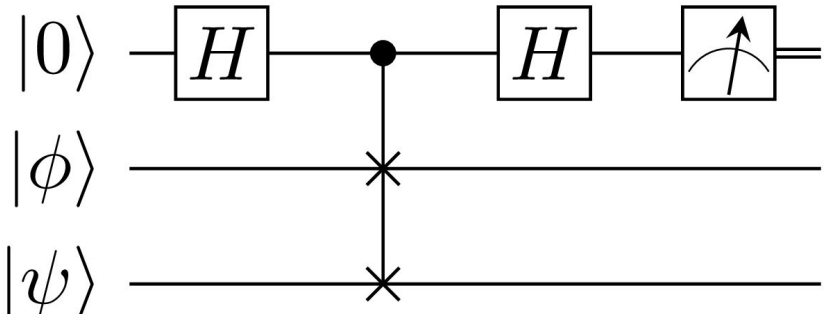
3. Quantum Cosine Classifier

Swap Test

- Given two quantum states, $|\phi\rangle$ and $|\psi\rangle$, the circuit outputs a Bernoulli random variable that is 0 with probability $\frac{1}{2} + \frac{1}{2}|\langle\phi|\psi\rangle|^2$



Swap Test Circuit



1. After the 1st Hadamard gate: $\frac{1}{\sqrt{2}}(|0,\phi,\psi\rangle + |1,\phi,\psi\rangle)$.
2. After the controlled swap gate: $\frac{1}{\sqrt{2}}(|0,\phi,\psi\rangle + |1,\psi,\phi\rangle)$.
3. After the 2nd Hadamard gate:

$$\frac{1}{2}(|0,\phi,\psi\rangle + |1,\phi,\psi\rangle + |0,\psi,\phi\rangle - |1,\psi,\phi\rangle) = \frac{1}{2}|0\rangle(|\phi,\psi\rangle + |\psi,\phi\rangle) + \frac{1}{2}|1\rangle(|\phi,\psi\rangle - |\psi,\phi\rangle)$$

4. Measurement:

$$P(\text{First qubit} = 0) = \frac{1}{2}\Big(\langle\phi|\langle\psi| + \langle\psi|\langle\phi|\Big)\frac{1}{2}\Big(|\phi\rangle|\psi\rangle + |\psi\rangle|\phi\rangle\Big) = \frac{1}{2} + \frac{1}{2}|\langle\psi|\phi\rangle|^2$$

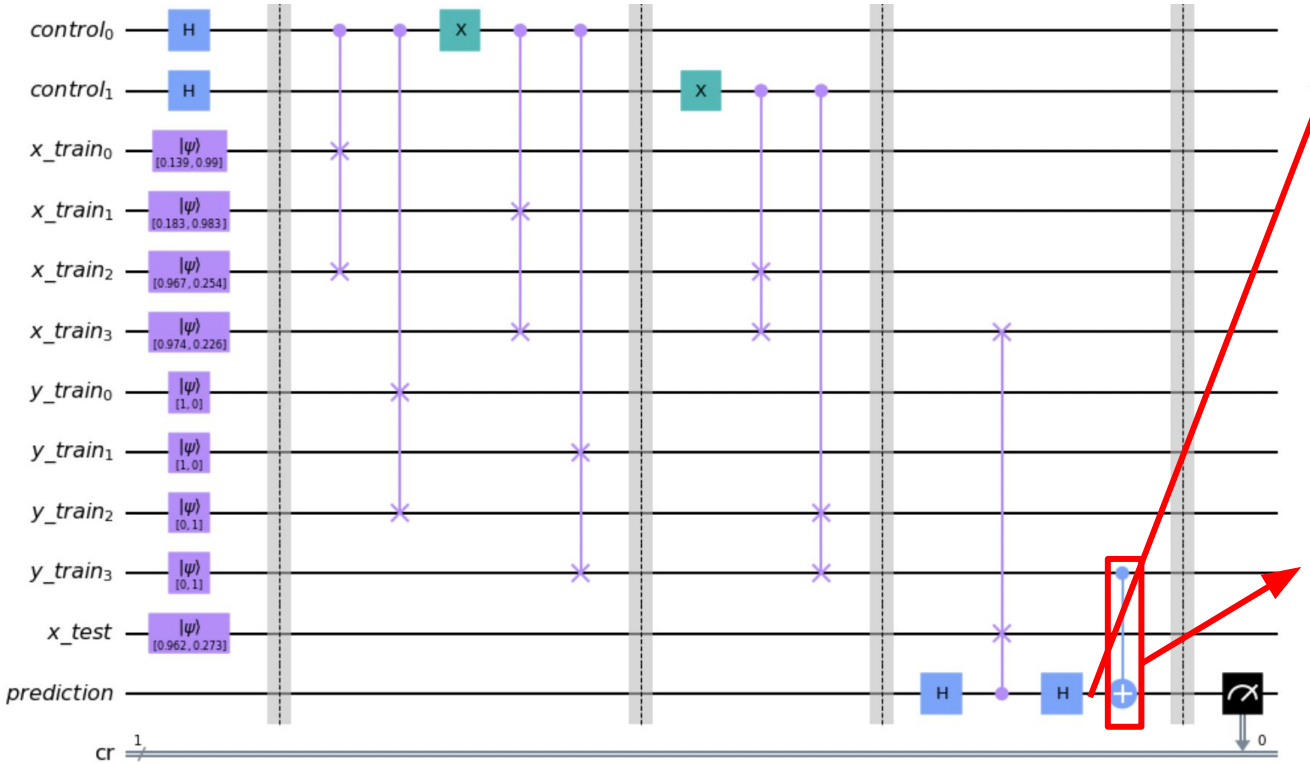
3. Quantum Cosine Classifier

Add the swap test to the quantum ensemble circuit

Perform a swap test between the last training sample and the testing sample

3. Quantum Cosine Classifier

Add the swap test to the quantum ensemble circuit



P(prediction = 0) = ½ + ½ $|\langle x\_train \mid x\_test \rangle|^2$

Cosine classifier output:
P(prediction = y_train) = ½ + ½ $|\langle x\_train | x\_test \rangle|^2$

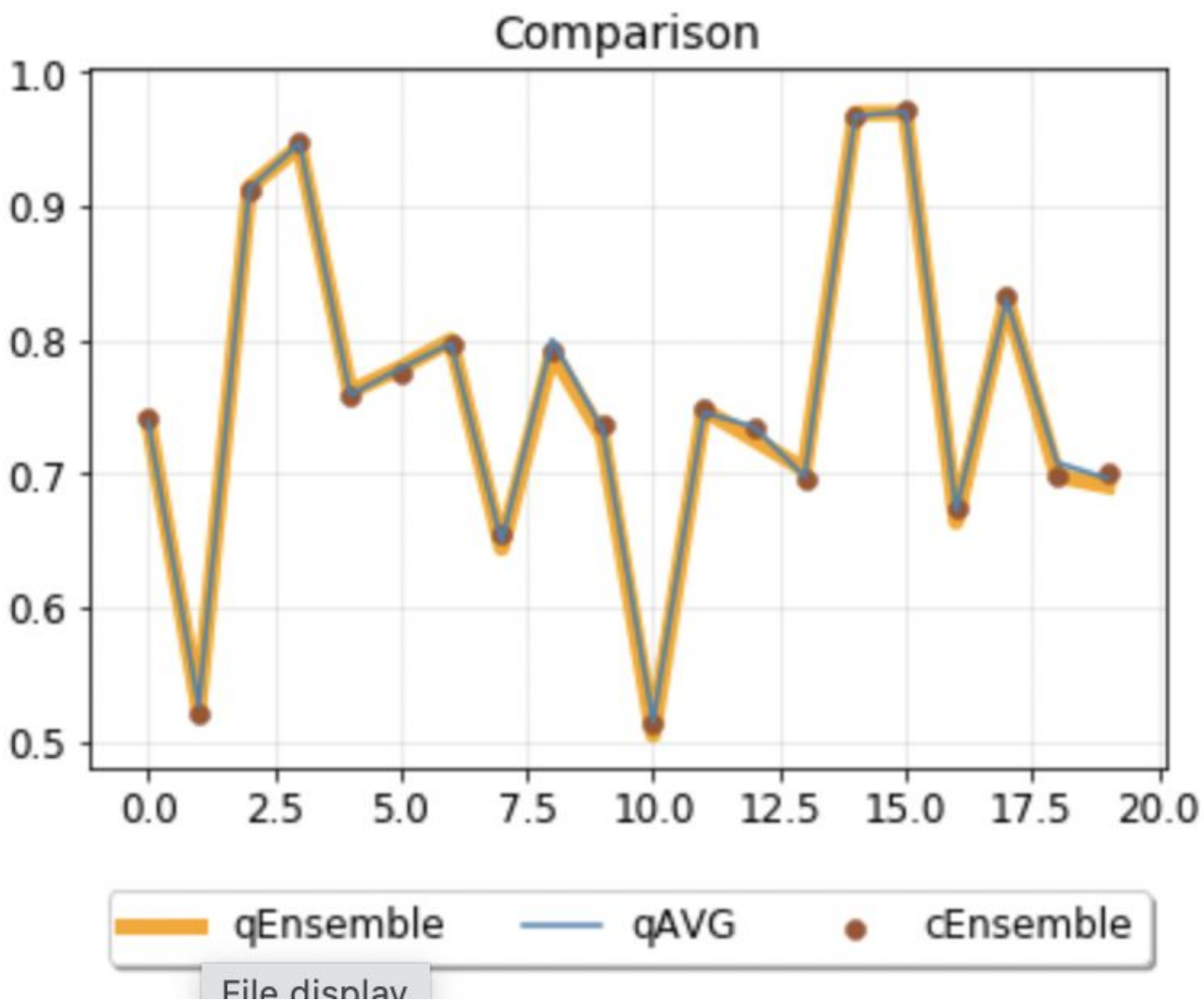
Controlled NOT gate:

- y_train = 0, do nothing
- y_train = 1, invert the prediction bit

4. Measurement

- Once the *Learning via interference* stage successfully encodes the probability of the test data belonging to either class 0 or 1 into the output test qubit, we make our prediction by simply measuring the prediction qubit (in the z-basis).
- The expectation of this measurement is precisely the average prediction over the 2^d classifiers:

$$\langle M \rangle = \langle \psi | I^{\otimes d} \otimes I \otimes I \otimes M | \psi \rangle$$
$$= \frac{1}{2^d} \sum_{b=1}^{2^d} \langle \widehat{f_b} | M | \widehat{f_b} \rangle = \frac{1}{2^d} \sum_{b=1}^{2^d} \langle M_b \rangle$$
$$= \frac{1}{2^d} \sum_{b=1}^{2^d} \widehat{f_b}$$


Comparison
1.0
0.9
0.8
0.7
0.6
0.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
qEnsemble
qAVG
cEnsemble

Recap of algorithm:

Algorithm 1: Quantum ensemble of quantum cosine classifiers

Result: Predictions of the binary target value for all points in the test set

Input:
- $2n$–qubit data register, d–qubit control register, 2-qubit test register
- Pauli-Z (measurement) operator $\langle \sigma_z \rangle$

for *each point* $\tilde{x}$ *in the test set* **do**

 # (Step 1) State Preparation

Encode n random training points into the $n \times 2$ qubits of the $data$ register:

$(x_1, y_1), \ldots, (x_n, y_n) \xrightarrow{S_{(x,y)}} |x_1, \ldots, x_n;\ y_1, \ldots, y_n\rangle = |features;\ labels\rangle$

Initialise the d qubits of $control$ register into a uniform superposition: $|0 \ldots 0\rangle \xrightarrow{W} \frac{1}{\sqrt{2^d}} \sum_{k=0}^{2^d-1} |k\rangle$

Initialise the $test$ register: $|0, 0\rangle \xrightarrow{S_{(\tilde{x},0)}} |\tilde{x}, 0\rangle$

 # (Step 2) Sampling in superposition

for *each qubit in the control register* $(i = 1, \ldots d)$ **do**

Select two pairs of random integers l, m and l', m' between 1 and n;

C-SWAP$(control(i), features(l), features(m))$;
C-SWAP$(control(i), labels(l), labels(m))$;

Apply Pauli-X gate to the current $control$ qubit ;

C-SWAP$\left(control(i), features\left(l'\right), features\left(m'\right)\right)$;
C-SWAP$\left(control(i), labels\left(l'\right), labels\left(m'\right)\right)$;

end

 # (Step 3) Learning via Interference

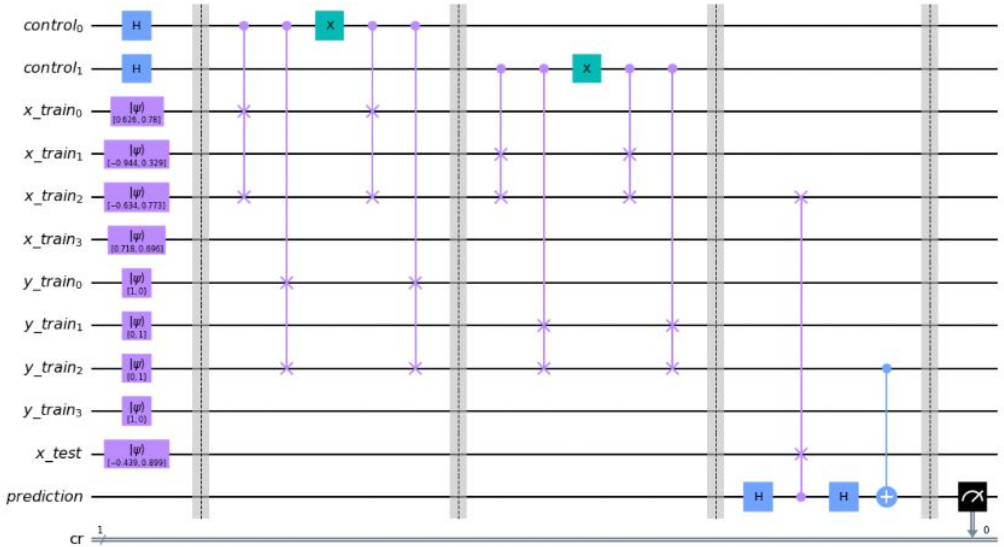
Apply the quantum cosine classifier (gate F) using as training set a random pair of qubits (*features, labels*) of the $data$ register;

 # (Step 4) Measurement

Measure the $test$ register using $\langle \sigma_z \rangle$ operator

end

Output: Ensemble predictions for all points in the test set;



Complexity advantages of quantum ensemble

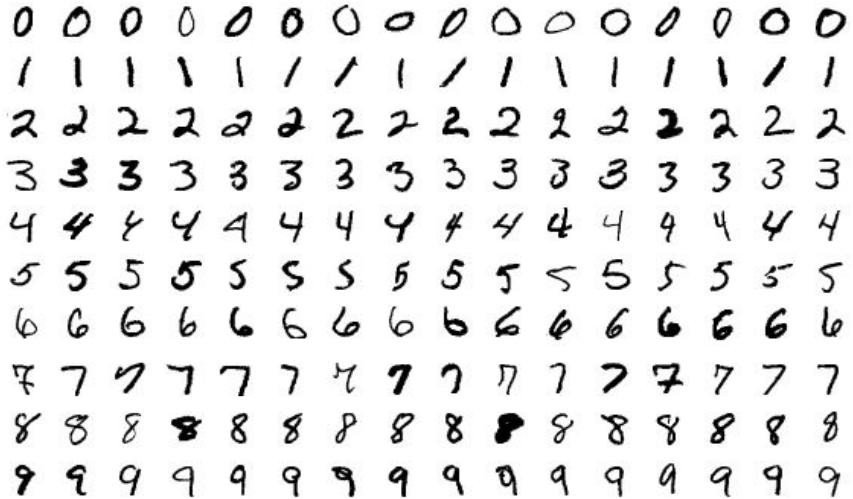
- Classically, given B base models, N p-dimensional training data:

$$\underbrace{\mathcal{O}(BN^{\alpha}p^{\beta})}_{\text{Training}} + \underbrace{\mathcal{O}(Bp)}_{\text{Testing}} \qquad \alpha, \beta \geq 1,$$

- Quantum ensemble complexity depends on encoding scheme S(x,y) and unitaries U(i,j), which depends on data encoding. However, the advantage is generally 3-fold:
  - Generates 2^d trajectories/models in d steps.
  - Single execution of cosine classification is simultaneously propagated across all trajectories
  - Elimination of the averaging step

Experiments

- Experimented on Digits MNIST and Fashion MNIST datasets



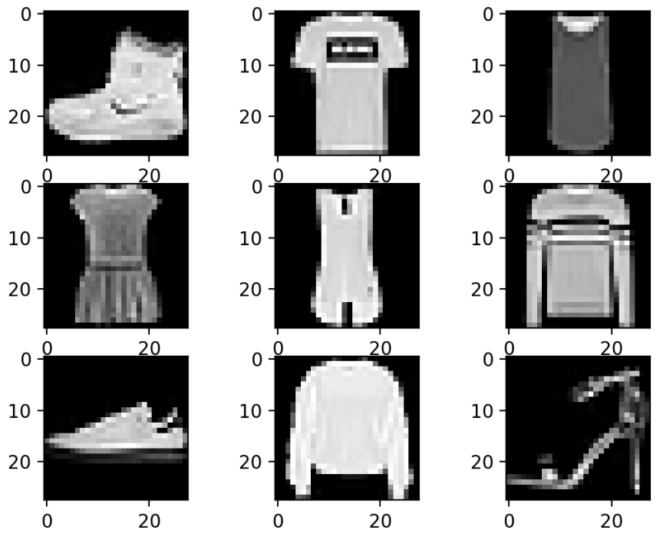
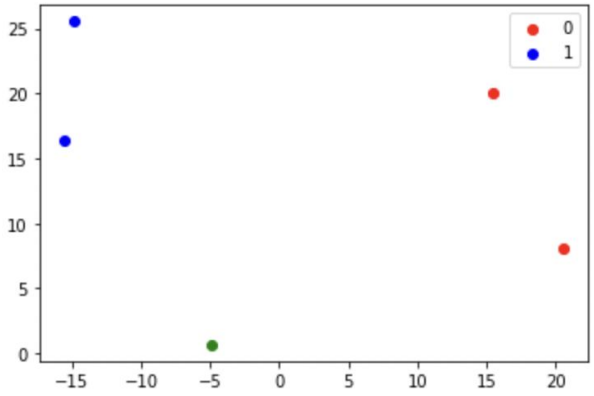


Table: Accuracies as a function of circuit depth d and size of training sample N

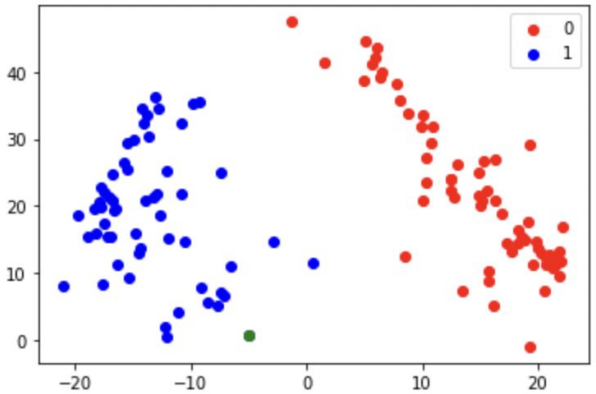
d	N	Digits MNIST (0 vs 9)		Fashion MNIST (T-shirt vs Ankle boot)	
		Mean	Std	Mean	Std
0	1	0.49	0.050	0.53	0.038
1	2	0.60	0.027	0.61	0.035
2	4	0.88	0.145	0.89	0.069
3	8	0.9	0.102	0.91	0.042

Example:

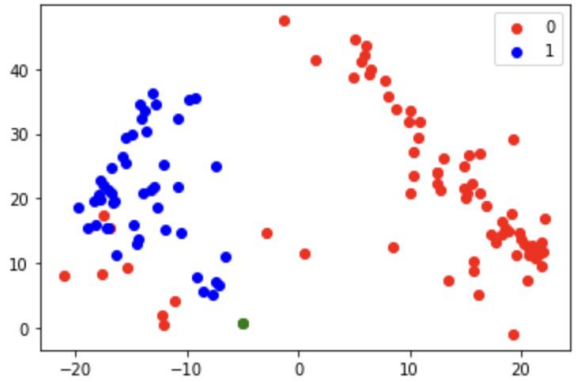
Training data:



Correct labels of test set:



Predictions:



Limitations of current model + extensions to the original paper:

Original paper:

- Supports only 2-dimensional feature input, hence requires PCA preprocessing → Developed higher dimensional data encoding.
- Cosine metric requires a little further preprocessing (i.e. shifting entire dataset up). → Developed L2 distance classifier.
- Non-parametric circuit.
- Predicts on test data one at a time → Can potentially parallelize predictions the same way sampling of training data was parallelized.

Extensions to original paper

N-dimensional data encoding

Amplitude embedding

A normalized classical N-dimensional ($N = 2^n$) data point $x = (x_1, \ldots, x_N)$ can be represented by the amplitudes of a n-qubit quantum state $|\psi_x\rangle$

$$|\psi_x\rangle = \sum_{i=1}^{N} x_i |i\rangle$$

e.g. $x = (½, ½, 0, 1/(2^{½}))$

$$|\psi_x\rangle = \tfrac{1}{2}|00\rangle + \tfrac{1}{2}|01\rangle + 0\,|10\rangle + \tfrac{1}{\sqrt{2}}|11\rangle$$

L2 Distance Classifier

- Given two **normalized** N-dimensional vectors x and y
- L2 / Euclidean distance between x and y is

$$d(x,\ y)\ = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

- $0 \leq d(x, y) \leq 2$, so $0 \leq (d(x, y))^2 \leq 4$
- Given a training data point, $(x_b,\ y_b)$ and a testing data point, $x^{(test)}$, we define the L2 Distance classifier as follows:

$$y^{(\text{test})} = \begin{cases} y_b & \frac{1}{4}(d(x^{(\text{test})}, x_b))^2 \leq \frac{1}{2} \\ 1 - y_b & \frac{1}{4}(d(x^{(\text{test})}, x_b))^2 > \frac{1}{2} \end{cases}$$

L2 Distance Classifier

- Encode two 2^n-dimensional vectors x_a and x_b into two n-bit quantum states, $|x_a\rangle$ and $|x_b\rangle$. The circuit outputs $(d(x_a,x_b))^2/(2Z)$, where $Z = |x_a|^2 + |x_b|^2$
- Prepare an ancilla qubit and entangle it with $|x_a\rangle$ and $|x_b\rangle$, such that the state of the whole system become:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0, x_a\rangle + |1, x_b\rangle)$$

- Measure the ancilla qubit in the following basis:

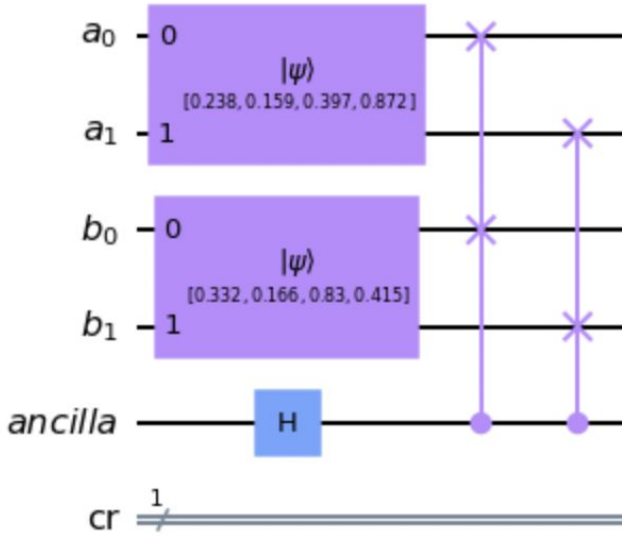
$$|\phi\rangle = \frac{1}{\sqrt{Z}}(|\overrightarrow{x_a}||0\rangle - |\vec{x}_b||1\rangle)$$

- It can be verified that the probability of getting the ancilla in the $|\phi\rangle$ state is $(d(x_a,x_b))^2/(2Z)$.

L2 Distance Classifier

How to prepare $|\psi\rangle = \frac{1}{\sqrt{2}}(|0, x_a\rangle + |1, x_b\rangle)$

- Hadamard + CSWAP gate



L2 Distance Classifier

How to measure the ancilla in the following basis?

$$|\phi\rangle = \frac{1}{\sqrt{Z}}(|\overrightarrow{x_a}||0\rangle - |\vec{x}_b||1\rangle)$$

- Currently, we assume that x_a and x_b are normalized. Thus $|\phi\rangle = |->$. Therefore, we only need to measure in the $|+\rangle$ and $|->$ basis.
- If x_a and x_b are NOT normalized, in our quantum ensemble scenario where many pairs of samples are sharing the same ancilla, the projective measurement required for different pairs of samples may be different.

L2 Distance Classifier

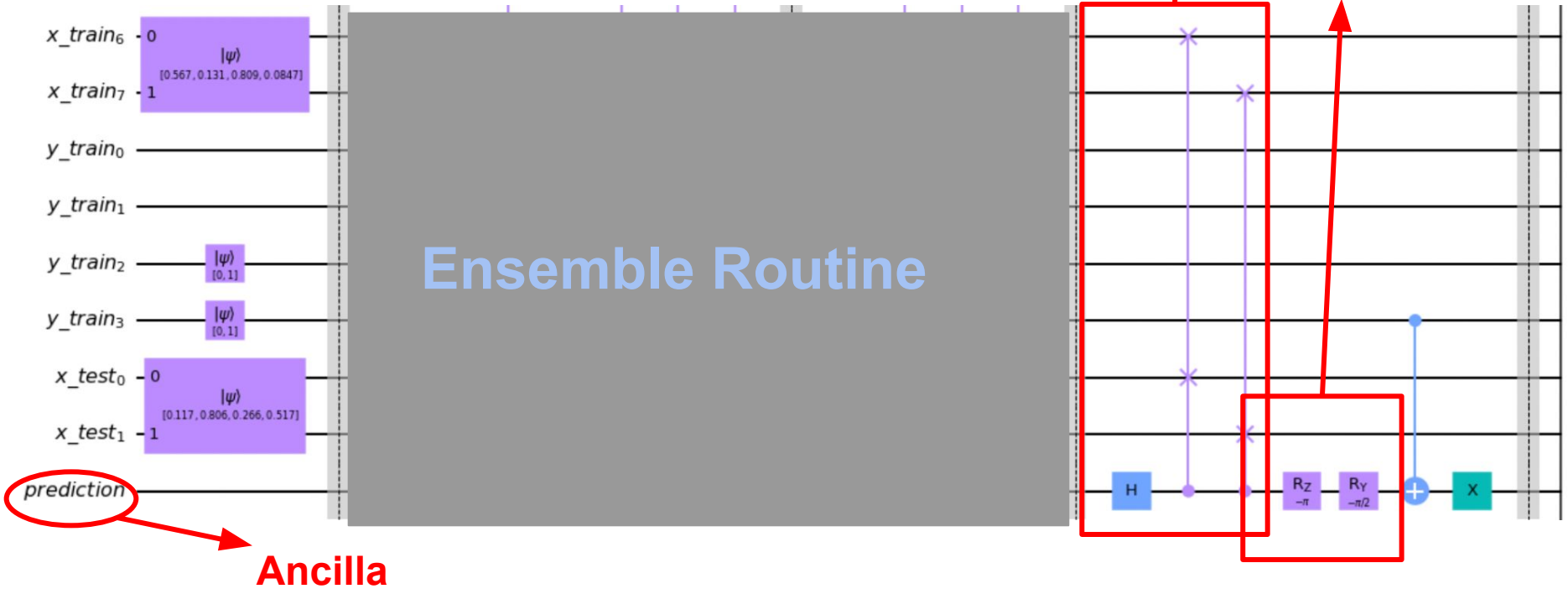
- the probability of getting the ancilla in the $|\phi>$ state is $(d(x_a,x_b))^2/(2Z)$
- We want the probability of getting the ancilla in the $|\phi>$ state (the new |0> state) and $|\phi^T>$ state (the new |1> state) to be related to y_a
- **Apply a CNOT gate to the ancilla, controlled by y_a**
- Assume that the closer the L2 distance between x_a and x_b, the more likely that $y_a = y_b$. **Distance is negatively correlated with probability. Thus, we need to apply an X gate to the ancilla.**

L2 Distance Classifier

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0, x_a\rangle + |1, x_b\rangle)$$

Circuit

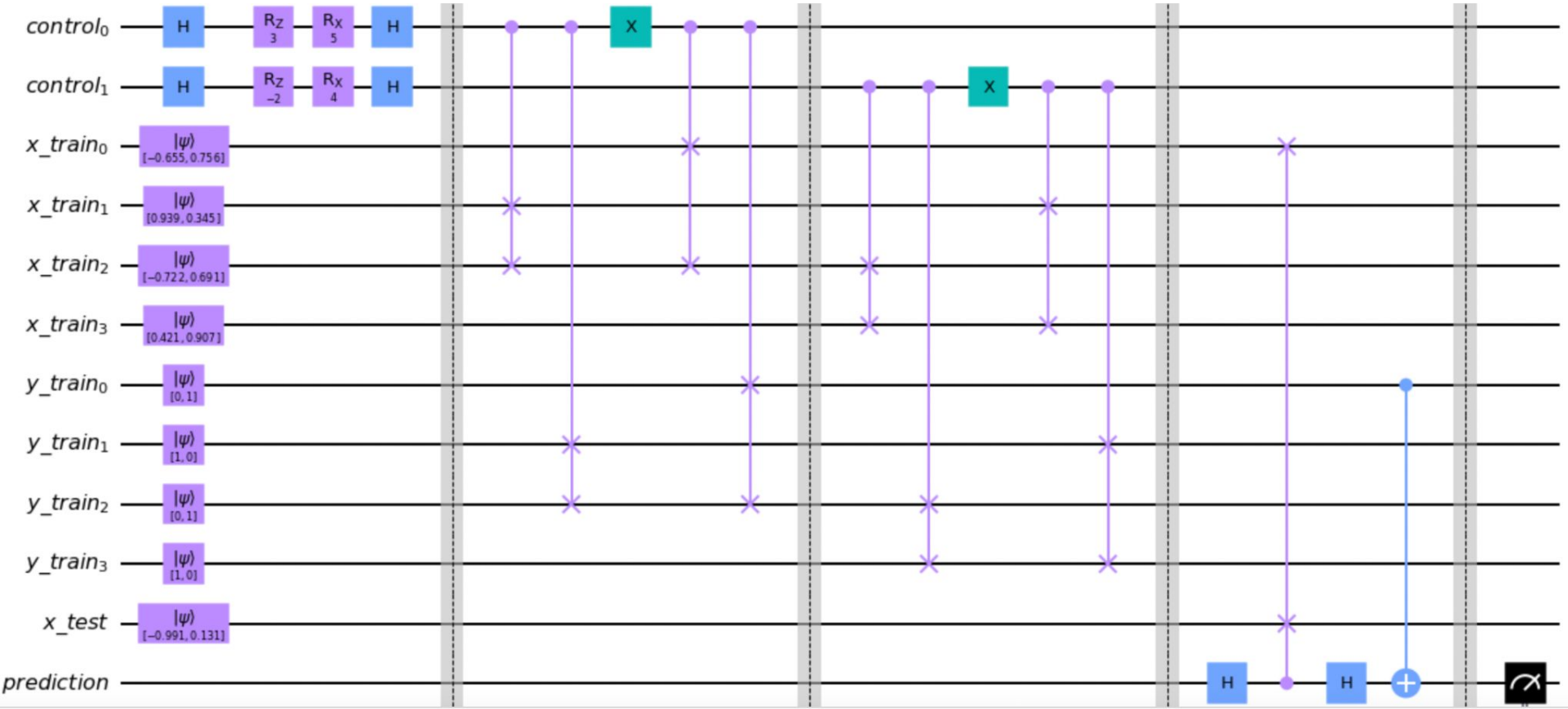
$$|\phi\rangle = \frac{1}{\sqrt{Z}}(|\overrightarrow{x_a}||0\rangle - |\vec{x}_b||1\rangle)$$



Quantum boosting methods (WIP)

- Idea: Instead of sampling from each training data uniformly, classical boosting (e.g. AdaBoost) maintains a probability distribution over training data and updates the distribution to **emphasize misclassified data**. The next classifier to enter the ensemble is trained based on the updated distribution.
- Adding classifiers one at a time to the ensemble defeats the point of parallelism. Instead, just have the circuit choose for itself which classifiers to emphasize. → Parametrize rotations on control registers and apply gradient descent against parameters with respect to loss function.

$$|\Phi_f\rangle = \frac{1}{\sqrt{2^d}} \sum_{b=1}^{2^d} \alpha_b \, |b\rangle \, |\hat{f}_b\rangle$$



Recall: In general, we can also parameterize the U(i,j)'s to enable greater expressivity/flexible feature transformations for each subclassifier.

Acknowledgements

Macaluso, Antonio, et al. “Quantum Ensemble for Classification.” *ArXiv.org*, 18 Jan. 2022, https://arxiv.org/abs/2007.01028.

Lloyd, Seth, et al. “Quantum Algorithms for Supervised and Unsupervised Machine Learning.” *ArXiv.org*, 4 Nov. 2013, https://arxiv.org/abs/1307.0411.

Schuld, M., et al. “An Introduction to Quantum Machine Learning.” *ArXiv.org*, 10 Sept. 2014, https://arxiv.org/abs/1409.3097.

“Variational Classifier.” *PennyLane*, https://pennylane.ai/qml/demos/tutorial_variational_classifier.html.

Summary of data encoding methods:
https://medium.datadriveninvestor.com/all-about-data-encoding-for-quantum-machine-learning-2a7344b1dfef

Custom gates and parameterization in Qiskit:
https://qiskit.org/documentation/tutorials/circuits_advanced/01_advanced_circuits.html#Parameterized-circuits

Gradients in qiskit: https://qiskit.org/documentation/tutorials/operators/02_gradients_framework.html

Thank you!

L2 Distance classifier: Details on generating psi and phi

- Generating phi and estimating Z:

$$H = (|x_a||0\rangle\langle 0| + |x_b||1\rangle\langle 1|)\otimes\sigma_x$$

$$|\Omega\rangle := e^{-iHt}|-\rangle\otimes|0\rangle$$
$$= \frac{1}{\sqrt{2}}(\cos(|x_a|t)\,|0\rangle - \cos(|x_b|t)\,|1\rangle)\otimes|0\rangle - \frac{i}{\sqrt{2}}(\sin(|x_a|t)\,|0\rangle$$
$$- \sin(|x_b|t)\,|1\rangle)\otimes|1\rangle$$

For small t, apply small-angle approximation and measure ancilla to obtain |phi> with probability:

L2 distance classifier

- Our classifier is thus:

$$y^{test} = \begin{cases} y_b, & |\langle\psi|\phi\rangle|^2 \leq \frac{1}{2} \\ 1 - y_b, & |\langle\psi|\phi\rangle|^2 > \frac{1}{2} \end{cases}$$

This is in a form very similar to the cosine classifier, so we implement it in a similar way: