BSc Project:
Entanglement estimation of pure two-qubit states using artificial neural networks

Kin Ian Lo

February 23, 2019

Supervisor: Dr Florian Mintert
Assessor: Dr Rick Mukherjee
Word count: 5998

Abstract

In quantum information experiments, it is important to experimentally verify the existence of the desired entanglement in prepared states. One way to confirm the existence of entanglement is to estimate the entropy of entanglement, which is a quantitative measure of pure bipartite entanglement, from experimental data. Quantum state tomography (QST) aims to reconstruct the full quantum state through measurements. However, the determination of entanglement does not require the full state. QST is thus inefficient for entanglement estimation since it produces extra information that are irrelevant to the entropy of entanglement. In this work, we modified the Bayesian mean estimation (BME) method in QST for the specific purpose of entanglement estimation. We showed that this modified method gives the best accuracy possible in term of the mean squared error. We also tried to apply artificial neural networks on entanglement estimation and achieved accuracy approaching the accuracy of the modified BME.

Contents

Summary of work undertaken

Entropy of entanglement is the standard quantitative measure of entanglement for pure two-qubit state, i.e., a pair of 2-level quantum systems such as a pair of spin-1/2 particles. The aim of this project is to estimate the entropy of entanglement exists in a pure two-qubit state. We considered a measurement scheme in which the spin of $3N$ identical copies of the state are measured along three orthogonal directions. The accuracy of the estimation was quantified by the RMS error. We derived an algorithm to evaluate the minimum error for a given N using Bayesian inference. We later applied artificial neural network on entanglement estimation. Our numerical results showed that the best trained networks approached close to the predicted minimum error. The code for our implementation can be found in [].

Declaration of work undertaken

K. I. Lo and his project partner T. Yiu contributed equally to the project. K. I. Lo focused primarily on the theoretical work on Bayesian mean estimation and made the main contribution to the proof and the algorithm in Appendix A. T. Yiu focused more on the architecture design and the hyper-parameter optimisation of artificial neural networks.

1 Introduction

Quantum entanglement is an unique feature of quantum mechanics not found in classical physics. In 1935, Einstein, Podolsky, and Rosen (EPR) noticed the non-local instantaneous "spooky action at a distance" and raised the EPR paradox in an attempt to argue that quantum mechanics was not a complete theory of reality [1]. Bell's paper in 1964 proposed the Bell's inequality which allowed statistical verification quantum entanglement. In contrast to Bell's belief, later experiments [2, 3] confirmed the existence of quantum entanglement. Moreover, Bell's results also provided the first quantitative description of entanglement. Since then, entanglement has been a central topic to the quantum information theory as many applications of entanglement has been proposed: quantum teleportation [4], quantum cryptography [5], quantum error correction [5–7] and quantum computation [8–10]. In the actual building of quantum devices, it is crucial to reliably confirm that the prepare quantum states have the desired properties, e.g. entanglement. Entanglement estimation aims to estimate the entanglement of prepared states with measurements. The entropy of entanglement is the standard measure of entanglement for pure bipartite states and this measure will be used throughout this report. In this project, we focused on the entanglement estimation for the simple case of pure two-qubit state.

Quantum state tomography (QST), in contrast to entanglement estimation, aims to estimate the full quantum state through measurements. Nonetheless, in certain applications, only the entanglement of the prepared states are of interests. It is of course possible to obtain the entanglement from the full state estimated by QST. However, the determination of entanglement do not require the full information about the state. QST thus produces unnecessary information irrelevant to entanglement estimation. Nonetheless, the framework and techniques in QST certainly sheds light on entanglement estimation. In this project, we tried to make modification to the existing methods in QST for the purpose of entanglement estimation. We will also use the measurement scheme often found in QST [11], that is, a complete set of projective spin-1/2 measurements.

Artificial neural network (ANN) is a versatile data-driven machine learning algorithm what is gaining popularity recently. ANNs learn to approximate a desired function by being fed many examples of input-output pairs of the function. Recently, ANN was used for QST [12], this inspired us to consider ANN applied on entanglement estimation. This report is divided in to 4 parts. We first introduce the basics of quantum information. Secondly, We review the methods in QST and adapt them in entanglement estimation. Thirdly, we investigate the possibility of applying artificial neural network (ANN) in entanglement estimation. Lastly, we compare the discussed methods by their root mean squared (RMS) error:

$$\text{RMS error} \equiv \sqrt{\langle (S_{\text{true}} - S_{\text{est}})^2 \rangle} \equiv \sqrt{\text{MS error}} \tag{1.1}$$

where S_{est} is the estimation and S_{true} is the true value. The expectation is taken over a distribution of states $\rho(\mathbf{r})$ for which we will define later. We will often discuss RMS in terms of the mean squared (MS) error for notational convenience.

2 Basics of quantum information

The following sections introduce the necessary background in quantum information for this project. Most of the derivations are adapted from the book by M. Nielsen and I. Chuang [13], unless otherwise stated.

2.1 Qubit and two-qubit state

A (pure) *qubit* is a two-level quantum system, i.e. its state vector $|\psi\rangle \in \mathcal{H}$ can be written as

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \qquad \text{where } \alpha, \beta \in \mathbb{C} \text{ and } |\alpha|^2 + |\beta|^2 = 1 \tag{2.1}$$

where $\{|0\rangle, |1\rangle\}$ forms a orthonormal basis of the Hilbert space $\mathcal{H}$. Suppose now we have two independent qubits, A and B, each described by their state vector $|\psi_A\rangle \in \mathcal{H}_A$ and $|\psi_B\rangle \in \mathcal{H}_B$ respectively. Explicitly,

$$|\psi_A\rangle = \alpha_A |0_A\rangle + \beta_A |1_A\rangle \quad \text{and} \quad |\psi_B\rangle = \alpha_B |0_B\rangle + \beta_B |1_B\rangle \tag{2.2}$$

We can choose to equivalently describe the two qubits together as a composite state vector $|\psi_{AB}\rangle \in \mathcal{H}_{AB} = \mathcal{H}_A \otimes \mathcal{H}_B$ with the tensor product:

$$\begin{aligned} |\psi_{AB}\rangle &= |\psi_A\rangle \otimes |\psi_B\rangle && (2.3a) \\ &= \alpha_A\alpha_B |0_A\rangle \otimes |0_B\rangle + \alpha_A\beta_B |0_A\rangle \otimes |1_B\rangle + \beta_A\alpha_B |1_A\rangle \otimes |0_B\rangle + \beta_A\beta_B |1_A\rangle \otimes |1_B\rangle && (2.3b) \\ &\equiv \alpha_A\alpha_B |00\rangle + \alpha_A\beta_B |01\rangle + \beta_A\alpha_B |10\rangle + \beta_A\beta_B |11\rangle && (2.3c) \end{aligned}$$

Since this composite state is composed from 2 qubits, it is called a *bipartite state*, or more specifically, a *two-qubit state*. The qubits composing the composite state are called the *subsystems*. We will always omit the tensor product symbol $\otimes$ and the subscripts A and B whenever there is no ambiguity by doing so.

The innocent looking equations in (2.2) actually tells us something very important: the two subsystems are *separable*, that is, not entangled. The ability to write a state vector for each subsystem is in fact the definition of a bipartite state being separable. Physically, this means that measuring one of the subsystems has no effect whatsoever on the other subsystem. If a bipartite state is separable, it is really a matter of choice whether we describe the two qubits individually with their own state vectors ($|\psi_A\rangle$, $|\psi_B\rangle$) or with the tensor product state vector $|\psi_{AB}\rangle$. However, if we allow interactions between the subsystems, sometimes it is not possible to write $|\psi_{AB}\rangle$ as a tensor product. For instance, consider the singlet spin state

$$|\psi_{AB}\rangle = \frac{1}{\sqrt{2}}(|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle) \tag{2.4}$$

where we have opted for the arrow notations in place of the $|0\rangle$ and $|1\rangle$ notations. In this report, we will predominately use the arrow notation to make the discussion more relatable to actual experiments. One can easily show that there do not exist $|\psi_A\rangle$ and $|\psi_B\rangle$ such that $|\psi_{AB}\rangle = |\psi_A\rangle \otimes |\psi_B\rangle$. Therefore, we are forced to describe the bipartite state as a whole. This bipartite state is then called *entangled*. Entangled states have the characteristic that the measurements made separately on the two subsystems possess some kind of statistical correlations.

2.2 The need for quantifying entanglement

Although the question "Is this state entangled?" is a legitimate one, it is rather limited. To see why, we first look at the entangled singlet spin state

$$\frac{1}{\sqrt{2}}(|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle) \tag{2.5}$$

Measuring the spin of the first subsystem will cause the superposition to collapse and thus the state of the second subsystem is known for certain. This bipartite state is thus *maximally* entangled. Consider now another entangled spin state:

$$\frac{1}{\sqrt{3}}(|\uparrow\downarrow\rangle - |\downarrow\uparrow\rangle + |\downarrow\downarrow\rangle) \tag{2.6}$$

Notice that if now the measured spin is 'down' for the first subsystem, we are no longer sure about the state of the second subsystem. It is because the bipartite state is collapsed into a superposition $(|\downarrow\uparrow\rangle - |\downarrow\downarrow\rangle)/\sqrt{2}$. Although the two examples of bipartite states are both entangled, the extents of entanglement are different: the second state is, roughly speaking, less entangled. It is thus important to quantify the extent of entanglement rather than just to qualitatively classify states into entangled and not entangled. The entropy of entanglement quantifies the entanglement of pure bipartite states. To introduce entropy of entanglement, we first need to introduce the notion of mixed state and density operator.

2.3 Pure states and mixed states

A pure state is a quantum state that can be fully described by a state vector $|\psi\rangle$. A mixed state is a statistical ensemble $\{p_j, |\psi_j\rangle\}_{j=1}^{n}$ of n pure states $\{|\psi_j\rangle\}_{j=1}^{n}$, where p_j is the probability that the actual state is $|\psi_j\rangle$. The probabilities sum to unity, i.e., $\sum_j p_j = 1$. In other words, if we regard a quantum state as a mixed state, we are not sure about which pure state the quantum state is in, but we know all the possible pure states the quantum state could be in and we know the corresponding probability for each possible pure state.

Although we will only consider pure two-qubit states in this project, the idea of mixed state becomes relevant when we try to describe a subsystem of a composite state, e.g., the first qubit in a two-qubit state.

2.4 Density operators

Density operator not only gives a convenient mathematical description of mixed state, it also eliminates any non-measurable information about a quantum state, be it pure or mixed. It is well known that the global phase of a state vector is non-measurable, i.e., $e^{i\phi}|\psi\rangle$ represents the same physical state as $|\psi\rangle$ for any real number ϕ. We will see that this non-measurable information is eliminated in the density operator. We first consider the case of pure state and will later generalise to mixed state. Consider an arbitrary pure state $|\psi\rangle$ and an arbitrary observable Q. The expectation of Q can be written as

$$\langle Q\rangle \equiv \langle\psi|Q|\psi\rangle \tag{2.7}$$

Now consider a orthonormal basis $\{|i\rangle\}$. Since the identity operator can be written as $\mathrm{I} = \sum_i |i\rangle\langle i|$, the above can be written as

$$\langle Q\rangle = \langle\psi|QI|\psi\rangle = \langle\psi|Q\left(\sum_i |i\rangle\langle i|\right)|\psi\rangle = \sum_i \langle\psi|Q|i\rangle\langle i|\psi\rangle = \sum_i \langle i|\psi\rangle\langle\psi|Q|i\rangle = \sum_i \langle i|\Big(|\psi\rangle\langle\psi|\,Q\Big)|i\rangle \tag{2.8}$$

By definition $\sum_i \langle i|A|i\rangle$ is the trace $\mathrm{Tr}\,A$ of the operator A, thus

$$\langle Q\rangle = \mathrm{Tr}\Big(|\psi\rangle\langle\psi|\,Q\Big) \equiv \mathrm{Tr}(\hat{\rho}_{\text{pure}}\,Q) \tag{2.9}$$

where we have defined the density operator (for pure state)

$$\hat{\rho}_{\text{pure}} = |\psi\rangle\langle\psi| \tag{2.10}$$

Notice that the expectation of any observable Q can be evaluated once $\hat{\rho}_{\text{pure}}$ is known. Moreover, the non-measurable global phase is eliminated in the density operator representation since $\hat{\rho}_{\text{pure}} = e^{i\phi}|\psi\rangle\langle\psi|e^{-i\phi} = |\psi\rangle\langle\psi|$. Therefore, the density operator indeed contains only the measurable information about the state and nothing else.

Now we generalise the idea to mixed states as it is most useful in this context. Consider a mixed state represented as an statistical ensemble $\{p_j, |\psi_j\rangle\}$. It should be obvious that the expectation of the observable Q can be written as a convex sum

$$\langle Q\rangle = \sum_j p_j\,\langle\psi_j|Q|\psi_j\rangle \tag{2.11}$$

Using the results in (2.9) and the linearity of trace, the expectation of Q becomes

$$\langle Q\rangle = \mathrm{Tr}(\hat{\rho}\,Q) \tag{2.12}$$

where we define the density operator for mixed state

$$\hat{\rho} = \sum_j p_j \left|\psi_j\middle\rangle\middle\langle\psi_j\right| \tag{2.13}$$

Note that $\hat{\rho}$ gives us all the information to predict the expectation of any observable Q, since $\langle Q\rangle = \mathrm{Tr}(\hat{\rho}Q)$. The density operator representation of mixed states thus contains only the measurable information about the state. Moreover, the density operator also allows us to conveniently write a mixed state as a single mathematical object. There are three important properties of density operator which are easily verifiable directly from the definition:

$$\textbf{Trace unity} \quad \mathrm{Tr}\,\hat{\rho} = 1 \tag{2.14a}$$

$$\textbf{Hermiticity} \quad \hat{\rho}^\dagger = \hat{\rho} \tag{2.14b}$$

$$\textbf{Positivity} \quad \forall\,|\phi\rangle \in \mathcal{H},\ \langle\phi|\rho|\phi\rangle \geq 0 \tag{2.14c}$$

2.5 Bloch sphere and Bloch vector

The previous section established the fact that the density operator of a quantum state contains all the measurable information. Here we try to take a closer look at the density operator $\hat{\rho}$ for a qubit and eventually provide a parameterised representation of $\hat{\rho}$.

It is customary to express density operators by matrices for easier manipulations. When a density operator is written in its matrix form (with respect to some orthonormal basis), it is called a *density matrix*. We start by writing $\hat{\rho}$ generally in matrix form with the basis $\{|\uparrow\rangle, |\downarrow\rangle\}$:

$$\hat{\rho} = \alpha\,|\uparrow\rangle\langle\uparrow| + \beta\,|\uparrow\rangle\langle\downarrow| + \gamma\,|\downarrow\rangle\langle\uparrow| + \delta\,|\downarrow\rangle\langle\downarrow| = \begin{pmatrix}\alpha & \beta\\ \gamma & \delta\end{pmatrix} \qquad \text{where } \alpha,\beta,\gamma,\delta \in \mathbb{C} \tag{2.15}$$

Since a density operator is Hermitian, we know that $\alpha, \delta \in \mathbb{R}$ and $\gamma = \beta^*$. Moreover, $\alpha + \delta = 1$ since a density operator has trace one. We can thus simplify the density matrix

$$\hat{\rho} = \begin{pmatrix}\alpha & \beta\\ \beta^* & 1-\alpha\end{pmatrix} \tag{2.16}$$

We can see that $\hat{\rho}$ is specified by 3 real numbers, since $\alpha \in \mathbb{R}$ and $\beta \in \mathbb{C}$. This motivates us to represent $\hat{\rho}$ with a Cartesian vector $\mathbf{r} = (x, y, z) \in \mathbb{R}^3$ by writing $\alpha = (1+z)/2$ and $\beta = (x - iy)/2$. The density matrix then becomes

$$\hat{\rho} = \frac{1}{2}\begin{pmatrix}1+z & x-iy\\ x+iy & 1-z\end{pmatrix} = \frac{1}{2}(\mathrm{I} + x\,\hat{\sigma}_x + y\,\hat{\sigma}_y + z\,\hat{\sigma}_z) = \frac{\mathrm{I} + \mathbf{r}\cdot\hat{\boldsymbol{\sigma}}}{2} \tag{2.17}$$

The motivation for the substitutions for α and β may seem unclear but they allow us to write the reduced density matrix in terms of the Pauli spin matrices and the vector $\mathbf{r}$, where the identity matrix and standard Pauli spin matrices are defined as:

$$\mathrm{I} = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix} \qquad \hat{\sigma}_x = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix} \qquad \hat{\sigma}_y = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix} \qquad \hat{\sigma}_z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix} \tag{2.18}$$

Here we have used the shorthand notation $\hat{\boldsymbol{\sigma}} = (\hat{\sigma}_x, \hat{\sigma}_y, \hat{\sigma}_z)$ which is by no mean a legitimate vector. It should be clear from this point that any density operator of a qubit can be uniquely represented by a vector $\mathbf{r}$ which is coined the name *Bloch vector*.

It can be shown that the eigenvalues $\lambda_\pm$ of the density operator can be written as

$$\lambda_\pm = \frac{1 \pm \|\mathbf{r}\|}{2} \equiv \frac{1 \pm r}{2} \tag{2.19}$$

where we have defined the *Bloch length* $r \equiv \|\mathbf{r}\|$. Since a density operator is positive, its eigenvalues must be non-negative. We conclude that $r \leq 1$. All physical Bloch vectors are therefore confined to live inside/on a unit sphere which we called the *Bloch sphere*. It can be shown that a pure qubit state has a Bloch length $r = 1$ and a mixed qubit state has a Bloch length $r < 1$.

2.6 Von Neumann entropy

Von Neumann entropy (vN entropy) [14] is a quantum extension of the classical Shannon entropy [15]. Shannon entropy quantifies the uncertainty about a statistical ensemble. Consider a discrete random variable X described by the statistical ensemble $\{p_j, x_j\}_{j=1}^{n}$, that is, the variable X has the probability p_j to become the outcome x_j. The Shannon entropy of the ensemble is then $-\sum_j p_j \log_2 p_j$. Consider now a mixed state $\hat{\rho}$ which has the eigendecomposition

$$\hat{\rho} = \sum_{j=1}^{n} \lambda_j \, |\phi_j\rangle\langle\phi_j| \tag{2.20}$$

where $\{\lambda_j\}$ are the eigenvalues and $\{|\phi_j\rangle\}$ are the corresponding eigenvectors. One can interpret the above as the mixed state having a underlying statistical ensemble $\{\lambda_j, |\phi_j\rangle\}$. However, this interpretation can neither be proved or disproved, since the underlying statistical ensemble is not measurable. Nonetheless, it can be shown that the ensemble $\{\lambda_j, |\phi_j\rangle\}$ gives the minimum Shannon entropy out of all possible underlying ensembles of $\hat{\rho}$. The vN entropy of the mixed state $S(\hat{\rho})$ is thus defined as the Shannon entropy of the ensemble $\{\lambda_j, |\phi_j\rangle\}$:

$$S(\hat{\rho}) = -\sum_{j=1}^{n} \lambda_j \log_2 \lambda_j = -\operatorname{Tr}(\hat{\rho} \log_2 \hat{\rho}) \tag{2.21}$$

The second equality can be shown by considering the power series of $\log_2(\hat{\rho})$. In this project, we will only consider the vN entropy of a qubit state so only the case of a qubit is consider here. It was given in (2.12) that the two eigenvalues of the density operator of a qubit are $\lambda_\pm = (1 \pm r)/2$, where $r \equiv \|\mathbf{r}\|$ is the length of the Bloch vector $\mathbf{r}$ associated to the density operator $\hat{\rho}$. The vN entropy, as a function of Bloch vector, of a mixed qubit state can be explicitly written as

$$S(\mathbf{r}) = S(r) = -\frac{1+r}{2} \log_2 \left(\frac{1+r}{2}\right) - \frac{1-r}{2} \log_2 \left(\frac{1-r}{2}\right) \tag{2.22}$$

The above implies that the vN entropy is completely determined by the Bloch length. This will be an important fact later in this report.

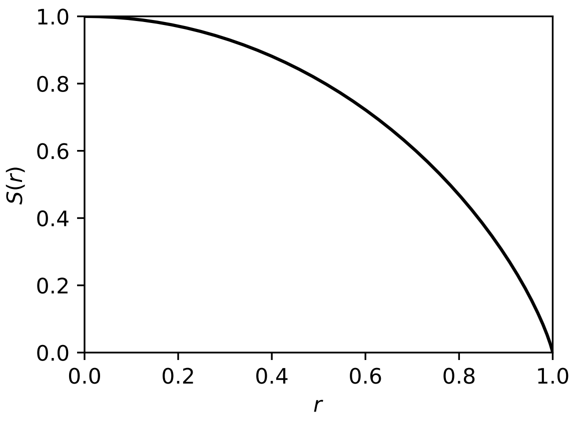


Figure 1: The relationship between vN entropy $S(r)$ and Bloch length r.

2.7 Reduced density operator

In this project, we will only consider local measurements in the sense that measurements are made on just one of the qubit in a two-qubit state. Since local measurements represent as a subset of all global measurements, the density operator would contain extra information that are not *locally* measurable. Thus, it is desirable to have a representation of a qubit in a two-qubit state. The representation should contain only locally measurable information and allow us to calculate the expectation of any local observable. The idea is to start from the density operator $\hat{\rho}_{AB} = |\Psi\rangle\langle\Psi|$ of a pure two-qubit state and then eliminate any information that are not locally measurable. Expanding $|\Psi\rangle\langle\Psi|$ with the basis $\{|i_A\rangle\}$ and $\{|j_B\rangle\}$ for the first and the second qubit respectively gives

$$|\Psi\rangle = \sum_{ij} \Psi_{ij} |i_A\rangle \otimes |j_B\rangle \tag{2.23}$$

Consider a local observable Q_A on the first qubit. The overall observable on the two-qubit system can be written as $Q = Q_A \otimes \mathrm{I}_B$, where I_B is the identity operator on the second qubit. The expectation of the overall observable is

$$\langle Q\rangle = \mathrm{Tr}(\hat{\rho}_{AB} Q) \tag{2.24}$$

$$\langle Q_A \otimes \mathrm{I}_B\rangle = \sum_{ij} \langle i_A| \otimes \langle j_B| \, \hat{\rho}_{AB}(Q_A \otimes \mathrm{I}_B) \, |i_A\rangle \otimes |j_B\rangle \tag{2.25}$$

$$\langle Q_A\rangle = \sum_{ij} \langle i_A| \, \langle j_B|\hat{\rho}_{AB}|j_B\rangle \, Q_A \, |i_A\rangle \tag{2.26}$$

$$\langle Q_A\rangle = \sum_{i} \langle i_A| \left(\sum_{j} \langle j_B|\hat{\rho}_{AB}|j_B\rangle \right) Q_A \, |i_A\rangle \equiv \mathrm{Tr}\left[\mathrm{Tr}_B(\hat{\rho}_{AB}) Q_A\right] \tag{2.27}$$

$$\langle Q_A\rangle = \mathrm{Tr}(\hat{\rho}_A Q_A) \tag{2.28}$$

where we define the *reduced density operator* of the subsystem A

$$\hat{\rho}_A = \mathrm{Tr}_B(\hat{\rho}_{AB}) \equiv \sum_{j} \langle j_B|\hat{\rho}_{AB}|j_B\rangle \tag{2.29}$$

and similarly for the subsystem B

$$\hat{\rho}_B = \mathrm{Tr}_A(\hat{\rho}_{AB}) \equiv \sum_i \langle i_A|\hat{\rho}_{AB}|i_A\rangle \tag{2.30}$$

It is obvious from the definition that $\hat{\rho}_A$ is Hermitian. Substituting Q_A for the identity I_A shows that $\hat{\rho}_A$ has trace one. Therefore, a reduced density operator can also be uniquely associated with a Bloch vector $\mathbf{r}$. Notice that $\hat{\rho}_A$ also allows us to determine the expectation of any local observable Q_A and thus $\hat{\rho}_A$ can be thought as a complete representation of the first qubit. It is therefore legitimate to treat the reduced density operator as the density operator of the first qubit when only local measurements are concerned.

2.8 Entropy of entanglement

The *entropy of entanglement* is a quantitative measure of entanglement for pure bipartite states. For a pure bipartite state $\hat{\rho}_{AB}$, the entropy of entanglement $E(\hat{\rho}_{AB})$ is just the vN entropy of either one of the subsystems [16]:

$$E(\hat{\rho}_{AB}) = S(\hat{\rho}_A) = S(\hat{\rho}_B) \tag{2.31}$$

The second equality may not be obvious but it can be shown that the vN entropy of the two subsystems are identical. This project aims to estimate the entropy of entanglement of pure two-qubit states from experimental data.

3 Adapting quantum state tomography

The aim of quantum state tomography (QST) is to reconstruct a quantum state through measurements. The Heisenberg uncertainty principle prevents the full determination of an arbitrary quantum state [17]. It is, however, possible to gain increasingly more information about the state by making more measurements. Since a quantum state is collapsed once being measured, measurements must be made on identical copies of the same state. However, the no-cloning theorem prevents us to perfectly replicate an unknown quantum state [18]. In fact, the no-cloning theorem also implies that perfect QST is impossible since a perfect QST allows one to prepare the exact same state. It is thus required that we are able to prepared the same state reliably [19].

The aim of this project is to estimate the entropy of entanglement of pure two-qubit states, i.e. the vN entropy of either one of the subsystems. Since the entropy of entanglement is fully determined by the reduced density operator of either subsystems, we will only consider local measurements on one of the subsystems. With QST, we could reconstruct the reduced density operator (and thus the Bloch vector) of one of the subsystems and then calculate the vN entropy with (2.21). However, QST produces extra information that are irrelevant to entanglement determination, e.g. the direction of the Bloch vector. Moreover, we will show that the best estimator of a state may not be the best estimator of its entanglement.

In the early development of QST, the scaled direct inversion (SDI) [20] was one of the first methods developed for single-qubit tomography. Later on, the more sophisticated *maximum likelihood estimation* (MLE) and *Bayesian mean estimation* (BME) [21] were developed based on the frequentist and Bayesian approaches respectively [22]. We will not consider MLE because it has been shown to be problematic (it produces non-physical density operator) and less accurate than BME [11]. In this project, we focus on SDI due to its simplicity and BME for its accuracy. In the following, we demonstrate how SDI and BME can be adapted for entanglement estimation.

3.1 Measurement scheme

In this project, we consider only projective spin-1/2 measurements on one of the subsystem of a two-qubit state. This type of measurements measure the spin of a spin-1/2 particle projected onto an axis along the direction of a unit vector $\mathbf{n} = (n_x, n_y, n_z)$. The operator of such a measurement can be expressed in terms of the Pauli matrices as $\hat{\sigma}_{\mathbf{n}} \equiv \mathbf{n} \cdot \hat{\boldsymbol{\sigma}} \equiv n_x \hat{\sigma}_x + n_y \hat{\sigma}_y + n_z \hat{\sigma}_z$ [11]. The measurements are made only on the first qubit $\hat{\rho}_A$ of the two-qubit state $\hat{\rho}_{AB}$. The expectation of $\hat{\sigma}_{\mathbf{n}}$ is given by (2.12):

$$\langle \hat{\sigma}_{\mathbf{n}} \rangle = \mathrm{Tr}\left(\hat{\rho}_A\, \hat{\sigma}_{\mathbf{n}}\right) = \mathrm{Tr}\left(\frac{\mathrm{I} + \mathbf{r} \cdot \hat{\boldsymbol{\sigma}}}{2} \mathbf{n} \cdot \hat{\boldsymbol{\sigma}}\right) = \mathbf{n} \cdot \mathbf{r} \tag{3.1}$$

where $\mathbf{r} = (x, y, z)$ is the Bloch vector associated to $\hat{\rho}_A$. The last equality can be easily verified by the commutation relations of the Pauli matrices and the fact that the Pauli matrices are traceless. It can be shown that $\hat{\sigma}_{\mathbf{n}}$ has eigenvalues $\{-1, 1\}$ and the associated eigenvectors $\{|{\downarrow_{\mathbf{n}}}\rangle, |{\uparrow_{\mathbf{n}}}\rangle\}$. The probability of getting the 'up' state $|{\uparrow_{\mathbf{n}}}\rangle$ and the probability of getting the 'down' state are [11]

$$p_{\uparrow\mathbf{n}} = \frac{1 + \mathbf{n} \cdot \mathbf{r}}{2} \quad \text{and} \quad p_{\downarrow\mathbf{n}} = \frac{1 - \mathbf{n} \cdot \mathbf{r}}{2} \tag{3.2}$$

The expectation of the observable $\hat{\sigma}_{\mathbf{n}}$ is then

$$\langle \hat{\sigma}_{\mathbf{n}} \rangle = (+1) p_{\uparrow\mathbf{n}} + (-1) p_{\downarrow\mathbf{n}} = 2\, p_{\uparrow\mathbf{n}} - 1 \tag{3.3}$$

In this scheme, a number of $N_{\mathbf{n}}$ two-qubit states are identically prepared, and the first qubit of each two-qubit state is measured by the observable $\hat{\sigma}_{\mathbf{n}}$. Out of the $N_{\mathbf{n}}$ measurements, $N_{\uparrow\mathbf{n}}$ of the outcomes are 'up' states and $N_{\downarrow\mathbf{n}} = N_{\mathbf{n}} - N_{\uparrow\mathbf{n}}$ are 'down' states. In order to fully determine the Bloch vector, we need to make measurements along three linearly independent directions. The natural choice of directions is a set of three orthogonal axes, labelled by $\mathbf{n} = \hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}$. The total number of measurements is thus $N_{\hat{\mathbf{x}}} + N_{\hat{\mathbf{y}}} + N_{\hat{\mathbf{z}}}$. For simplicity, the number of measures for each direction are set to be $N = N_{\hat{\mathbf{x}}} = N_{\hat{\mathbf{y}}} = N_{\hat{\mathbf{z}}}$ such that $3N$ measurements are made total. Since each spin-1/2 measurement can only give the 'up' or the 'down' state with certain fixed probabilities, the statistic should follows a binomial distribution for each axis of measurement. The probability of getting the number of 'up' state $N_{\uparrow\mathbf{n}}$ is

$$\mathcal{P}(N_{\uparrow\mathbf{n}}|\mathbf{r}) = \binom{N}{N_{\uparrow\mathbf{n}}} (p_{\uparrow\mathbf{n}})^{N_{\uparrow\mathbf{n}}} (p_{\downarrow\mathbf{n}})^{N - N_{\uparrow\mathbf{n}}} = \binom{N}{N_{\uparrow\mathbf{n}}} \left(\frac{1 + \mathbf{n} \cdot \mathbf{r}}{2}\right)^{N_{\uparrow\mathbf{n}}} \left(\frac{1 - \mathbf{n} \cdot \mathbf{r}}{2}\right)^{N - N_{\uparrow\mathbf{n}}} \tag{3.4}$$

The joint probability of getting the *experimental outcome* $\mathbf{N}_{\uparrow} = (N_{\hat{\mathbf{x}}\uparrow}, N_{\hat{\mathbf{y}}\uparrow}, N_{\hat{\mathbf{z}}\uparrow})$ is then

$$\mathcal{P}(\mathbf{N}_{\uparrow}|\mathbf{r}) = \prod_{\mathbf{n} = \hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}} \mathcal{P}(N_{\uparrow\mathbf{n}}|\mathbf{r}) = \prod_{\mathbf{n} = \hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}} \binom{N}{N_{\uparrow\mathbf{n}}} \left(\frac{1 + \mathbf{n} \cdot \mathbf{r}}{2}\right)^{N_{\uparrow\mathbf{n}}} \left(\frac{1 - \mathbf{n} \cdot \mathbf{r}}{2}\right)^{N - N_{\uparrow\mathbf{n}}} \tag{3.5}$$

Under this measurement scheme, the goal of QST is then to invert the above probability. There are other measurement schemes that adapts the measurement axis depending on previous measurement outcomes [23–25]. These adaptive schemes were experimentally shown to give better accuracy [26], but we shall not discuss them further.

3.2 The problem setting and assumptions

Now we have all the concepts and terminology introduced, we take the opportunity to state precisely the problem we are trying to solve in this project. We are given $3N$ identical copies of a pure two-qubit state $\hat{\rho}_{AB}$. The density operator $\hat{\rho}_{AB}$ was sampled from a distribution such that the Bloch vector $\mathbf{r}$ associated to the reduced density operator $\hat{\rho}_A$ follows a known distribution $\rho(\mathbf{r})\,\mathrm{d}\mathbf{r}$. The following discussion do not assume any particular form for the distribution $\rho(\mathbf{r})$ unless otherwise stated. However, for numerical results we have chosen to specialise on the volume-uniform distribution $\rho_{\mathrm{vol}}(\mathbf{r}) = 3/4\pi$ which is also called Hilbert-Schmidt-uniform [11]. We are then required to obtain an estimation $S_{\mathrm{est}}(\mathbf{N}_{\uparrow})$ on the vN entropy of $\hat{\rho}_A$ based the experimental outcome $\mathbf{N}_\uparrow$. We aim to minimise the MS error of our estimations. To compute the mean squared (MS) error, we sum over all the possible experimental outcomes and integrate over all physical Bloch vector:

$$\left\langle [S_{\text{est}}(\mathbf{N}_\uparrow) - S(\mathbf{r})]^2 \right\rangle = \sum_{\mathbf{N}_\uparrow} \int_{\|\mathbf{r}\|<1} [S_{\text{est}}(\mathbf{N}_\uparrow) - S(\mathbf{r})]^2 \rho(\mathbf{r}|\mathbf{N}_\uparrow)\mathcal{P}(\mathbf{N}_\uparrow)\, \mathrm{d}\mathbf{r} \tag{3.6}$$

where $\rho(\mathbf{r}|\mathbf{N}_\uparrow)\mathcal{P}(\mathbf{N}_\uparrow)\,\mathrm{d}\mathbf{r}$ is the joint probability for the experimental outcome to be $\mathbf{N}_\uparrow$ and the Bloch vector to lies within $\mathbf{r}$ and $\mathbf{r}+\mathrm{d}\mathbf{r}$. The prior probability $\mathcal{P}(\mathbf{N}_\uparrow)$ can be written as

$$\mathcal{P}(\mathbf{N}_\uparrow) = \int_{\|\mathbf{r}\|<1} \mathcal{P}(\mathbf{N}_\uparrow|\mathbf{r})\rho(\mathbf{r})\,\mathrm{d}\mathbf{r} \tag{3.7}$$

where $\mathcal{P}(\mathbf{N}_\uparrow|\mathbf{r})$ is given by (3.5). The posterior distribution of Bloch vector $\rho(\mathbf{r}|\mathbf{N}_\uparrow)$ is given by the Bayes' theorem

$$\rho(\mathbf{r}|\mathbf{N}_\uparrow) = \frac{\mathcal{P}(\mathbf{N}_\uparrow|\mathbf{r})\rho(\mathbf{r})}{\mathcal{P}(\mathbf{N}_\uparrow)} \tag{3.8}$$

In this report, we use the notation $\langle\cdot\rangle_{\mathbf{N}_\uparrow}$ for the expectation take oven the distribution $\rho(\mathbf{r}|\mathbf{N}_\uparrow)$, that is,

$$\langle\cdot\rangle_{\mathbf{N}_\uparrow} = \int_{\|\mathbf{r}\|<1} (\cdot)\rho(\mathbf{r}|\mathbf{N}_\uparrow)\,\mathrm{d}\mathbf{r} \tag{3.9}$$

With this notation, the MS error in (3.6) can be written as

$$\left\langle [S_{\text{est}}(\mathbf{N}_\uparrow) - S(\mathbf{r})]^2 \right\rangle = \sum_{\mathbf{N}_\uparrow} \mathcal{P}(\mathbf{N}_\uparrow) \left\langle (S_{\text{est}}(\mathbf{N}_\uparrow) - S(\mathbf{r}))^2 \right\rangle_{\mathbf{N}_\uparrow} \tag{3.10}$$

We will show how to actually evaluate the MS error later in this report.

3.3 Scaled direct inversion

Since $\langle\hat{\sigma}_\mathbf{n}\rangle = \mathbf{n}\cdot\mathbf{r}$ as given in (3.1), the Bloch vector can be written as the expectation of the Pauli matrices

$$\mathbf{r} = (x,y,z) = (\langle\hat{\sigma}_x\rangle, \langle\hat{\sigma}_y\rangle, \langle\hat{\sigma}_z\rangle) = 2(p_{\uparrow\hat{\mathbf{x}}}, p_{\uparrow\hat{\mathbf{y}}}, p_{\uparrow\hat{\mathbf{z}}}) - 1 \tag{3.11}$$

where the last equality is due to (3.3). The idea of direct inversion is to approximate the probabilities $p_{\uparrow\mathbf{n}}$ by the measured frequencies $N_{\uparrow\mathbf{n}}/N$. The Bloch vector estimated is then

$$\mathbf{r}_{\text{DI}} = 2\frac{(N_{\uparrow\hat{\mathbf{x}}}, N_{\uparrow\hat{\mathbf{y}}}, N_{\uparrow\hat{\mathbf{z}}})}{N} - 1 = 2\frac{\mathbf{N}_\uparrow}{N} - 1 \tag{3.12}$$

This procedure of inverting the experimental outcome $\mathbf{N}_\uparrow = (N_{\hat{\mathbf{x}}\uparrow}, N_{\hat{\mathbf{y}}\uparrow}, N_{\hat{\mathbf{z}}\uparrow})$ to the Bloch vector $\mathbf{r}_{\text{DI}}$ is called *direct inversion*, also called *linear inversion* in some literatures [23]. However, the length of the estimated Bloch vector can be greater than one. That means the measured density operator can be non-physical. An ad-hoc solution to this issue is to re-scale the estimated Bloch vector when its length is greater than one [11]. We define the *scaled* estimated Bloch vector

$$\mathbf{r}_{\text{SDI}} = \begin{cases} \mathbf{r}_{\text{DI}} & ,\|\mathbf{r}_{\text{DI}}\| \le 1 \\ \mathbf{r}_{\text{DI}}/\|\mathbf{r}_{\text{DI}}\| & ,\|\mathbf{r}_{\text{DI}}\| > 1 \end{cases} \tag{3.13}$$

This modification to direct inversion with the re-scaling is called the *scaled direct inversion* (SDI). We then calculate the vN entropy of $\mathbf{r}_{\text{SDI}}$ using (2.22)

$$S_{\text{est}}(\mathbf{N}_\uparrow) = S(\mathbf{r}_{\text{SDI}}) = -\frac{1+r_{\text{SDI}}}{2}\log_2\left(\frac{1+r_{\text{SDI}}}{2}\right) - \frac{1-r_{\text{SDI}}}{2}\log_2\left(\frac{1-r_{\text{SDI}}}{2}\right) \tag{3.14}$$

where $r_{\text{SDI}} \equiv \|\mathbf{r}_{\text{SDI}}\|$. The above equation summaries the method of SDI.

3.4 Bayesian mean estimation

Although SDI is easy to implement and conceptually easy to understand, it ignores the information given in the distribution of Bloch vector $\rho(\mathbf{r})$ completely. Now, we introduce the *Bayesian mean estimation* (BME) method in QST based on Bayesian inference [21] which make use of the distribution of Bloch vector.

In QST, the central idea of BME is that the mean of the Bloch vector *after* the measurements were made $\langle \mathbf{r} \rangle_{\mathbf{N}_\uparrow} = \int \mathbf{r}\,\rho(\mathbf{r}|\mathbf{N}_\uparrow)\,\mathrm{d}\mathbf{r}$ is the best estimator of the Bloch vector [22]. This is motivated by the fact that the fidelity (a measure of "closeness" between two states [27, 28]) is minimised the by the mean Bloch vector $\langle \mathbf{r} \rangle_{\mathbf{N}_\uparrow}$. Adapting this idea in QST, we propose that the best estimator of vN entropy is the mean entropy $\langle S(\mathbf{r}) \rangle_{\mathbf{N}_\uparrow} = \int S(\mathbf{r})\rho(\mathbf{r}|\mathbf{N}_\uparrow)\,\mathrm{d}\mathbf{r}$ based on the fact that the mean minimises the MS error in (3.10). To show the MS error is minimised, it is sufficed to show that every $\left\langle [S_{\text{est}}(\mathbf{N}_\uparrow) - S(\mathbf{r})]^2 \right\rangle_{\mathbf{N}_\uparrow}$ is minimised. It is trivial to show that

$$\left\langle [S_{\text{est}}(\mathbf{N}_\uparrow) - S(\mathbf{r})]^2 \right\rangle_{\mathbf{N}_\uparrow} = \left\langle S^2(\mathbf{r}) \right\rangle_{\mathbf{N}_\uparrow} - \langle S(\mathbf{r}) \rangle^2_{\mathbf{N}_\uparrow} + \Big[\langle S(\mathbf{r}) \rangle_{\mathbf{N}_\uparrow} - S_{\text{est}}(\mathbf{N}_\uparrow) \Big]^2 \tag{3.15}$$

One can see that the MS error is determined by a residue term $[\langle S(\mathbf{r}) \rangle_{\mathbf{N}_\uparrow} - S_{\text{est}}(\mathbf{N}_\uparrow)]^2$. It is obvious that MS error is minimised when this residue term becomes zero, that is, when $S_{\text{est}}(\mathbf{N}_\uparrow) = \langle S(\mathbf{r}) \rangle_{\mathbf{N}_\uparrow}$. Note that the minimum error $\left\langle S^2(\mathbf{r}) \right\rangle_{\mathbf{N}_\uparrow} - \langle S(\mathbf{r}) \rangle^2_{\mathbf{N}_\uparrow}$ is just the variance of of the posterior distribution of vN entropy $\rho(S|\mathbf{N}_\uparrow)$. Since the residue term is greater than zero for any estimator, we can conclude that BME is the best estimator in terms of RMS error. Furthermore, since $S\Big(\langle \mathbf{r} \rangle_{\mathbf{N}_\uparrow}\Big) \neq \langle S(\mathbf{r}) \rangle_{\mathbf{N}_\uparrow}$ in general, our modification to BME improves the accuracy for entanglement estimation under the same number of measurements N.

The implementation of BME lies in the evaluation the triple integral defining the mean vN entropy $\langle S(\mathbf{r}) \rangle_{\mathbf{N}_\uparrow}$. Note that the vN entropy is an isotropic function, i.e., the vN entropy only depends on the Bloch length. Here we use the opportunity to evaluate a more general case, the mean of any isotopic function $\langle f(r) \rangle_{\mathbf{N}_\uparrow}$:

$$\langle f(r) \rangle_{\mathbf{N}_\uparrow} = \int f(r)\rho(\mathbf{r}|\mathbf{N}_\uparrow)\,\mathrm{d}\mathbf{r} \tag{3.16}$$

Since the Bloch vector $\mathbf{r} \in \mathbb{R}^3$, this is a triple integral which is often difficult for numerical integration. By imposing the assumption that the distribution of Bloch vector is isotropic, i.e. $\rho(\mathbf{r})\,\mathrm{d}\mathbf{r} = \mathcal{R}(r)\,\mathrm{d}\mathbf{r}$ for some function of Bloch length $\mathcal{R}(r)$, we reduced the triple integral into a single integral which is much easier for numerical integrations. Since $f(r)$ has only radial dependence, it is customary to tackle the triple integral in spherical polar coordinate (r, θ, ϕ) such that $(x, y, z) = (r \sin\theta \cos\phi, r \sin\theta \sin\phi, r\cos\theta)$ with the Jacobian $r^2 \sin\theta$:

$$\langle f(r) \rangle_{\mathbf{N}_\uparrow} = \int_{r=0}^{1} \int_{\theta=0}^{\pi} \int_{\phi=0}^{2\pi} f(r)\rho(\mathbf{r}|\mathbf{N}_\uparrow) r^2 \sin\theta\,\mathrm{d}r\,\mathrm{d}\theta\,\mathrm{d}\phi \tag{3.17}$$

$$= \int_{r=0}^{1} f(r)\rho(r|\mathbf{N}_\uparrow)\,\mathrm{d}r \tag{3.18}$$

where we define the posterior distribution of Bloch length as

$$\rho(r|\mathbf{N}_\uparrow) = \int_{\theta=0}^{\pi} \int_{\phi=0}^{2\pi} \rho(\mathbf{r}|\mathbf{N}_\uparrow) r^2 \sin\theta\,\mathrm{d}\theta\,\mathrm{d}\phi \tag{3.19}$$

We proved that above is a product of $\mathcal{R}(r)$ and a polynomial in Bloch length r, along with a constant $\eta(\mathbf{N}_\uparrow)$:

$$\rho(r|\mathbf{N}_\uparrow) = \eta(\mathbf{N}_\uparrow)\mathcal{R}(r)\sum_{\beta=2}^{3N+2} B_\beta r^\beta \quad \text{and} \quad \eta(\mathbf{N}_\uparrow) = \frac{1}{2^{3N}}\frac{1}{\mathcal{P}(\mathbf{N}_\uparrow)}\binom{N}{N_{\hat{\mathbf{x}}\uparrow}}\binom{N}{N_{\hat{\mathbf{y}}\uparrow}}\binom{N}{N_{\hat{\mathbf{z}}\uparrow}} \tag{3.20}$$

where the coefficients B_β can be computed by a vectorised algorithm. For details on the proof and the algorithm, see Appendix A. In practice, the normalisation constant $\eta(\mathbf{N}_\uparrow)$ is computed by normalising $\mathcal{R}(r)\sum_\beta B_\beta r^\beta$. The prior probability $\mathcal{P}(\mathbf{N}_\uparrow)$ can be found subsequently from $\eta(\mathbf{N}_\uparrow)$. The mean $\langle f(r)\rangle_{\mathbf{N}_\uparrow}$ can then be written as

$$\langle f(r)\rangle_{\mathbf{N}_\uparrow} = \eta(\mathbf{N}_\uparrow)\int_{r=0}^{1} f(r)\mathcal{R}(r)\sum_{\beta=2}^{3N+2} B_\beta r^\beta \,\mathrm{d}r \tag{3.21}$$

which is often evaluated with numerical integration. With this result, we can evaluate the mean entropy $\langle S(\mathbf{r})\rangle_{\mathbf{N}_\uparrow}$, which is the estimation of vN entropy; the variance $\left\langle S^2(\mathbf{r})\right\rangle_{\mathbf{N}_\uparrow} - \langle S(\mathbf{r})\rangle^2_{\mathbf{N}_\uparrow}$, which is the minimum MS error; and the prior probability $\mathcal{P}(\mathbf{N}_\uparrow)$, which will be used to evaluate the overall MS error $\left\langle (S_{\text{est}}(\mathbf{N}_\uparrow) - S(\mathbf{r}))^2\right\rangle$ as given in (3.10).

4 Artificial neural networks

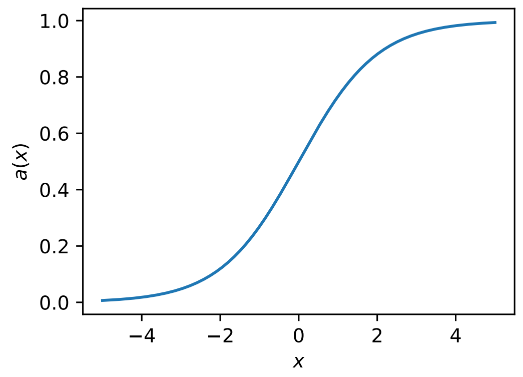
Artificial neural networks (ANNs) were first developed as a computational model of biological brains such as the one of human beings [29]. Traditional computing was done by manually instructing the computer how to react to possible inputs. However, sometimes it is not feasible to hard-code every instructions the computer should follow. An example is handwriting recognition. One could only imagine the difficulty to manually program a computer to convert an image of handwritten letters into text. On the other hand, ANNs learn by watching many examples of input and output pairs. Given enough examples, ANNs can generalise to other unseen inputs. ANN has found many applications that manual programming would be difficult, e.g. image recognition, speech recognition, financial forecasting, drug discovery, particle accelerator data analysis, etc [30]. Here we give a general introduction to ANN. For more details on ANN, see the report by the project partner [31]. The following introduction to ANN was adapted from the book by I. Goodfellow, Y. Bengio and C. Courville [29].

4.1 Introduction to neural networks

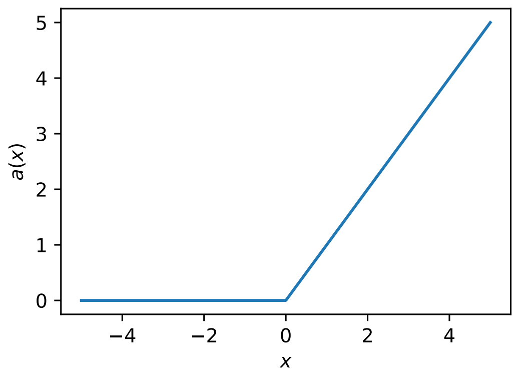
The most basis unit in a ANN is a *neuron* which a mathematical function $\hat{y} : \mathbb{R}^n \to \mathbb{R}$ that maps n_x inputs $\mathbf{x} = (x_1, x_2, ..., x_{n_x})$ into a single output y. The neuron has $n_x + 1$ tunable real parameters: the *weights* $\mathbf{w} = (w_1, w_2, ..., w_n)$ and the *bias* b. The neuron is also equipped with *activation function* $a : \mathbb{R} \to \mathbb{R}$. The output of a neuron is defined by a linear combination of the inputs with a bias b, followed by the activation function:

$$\hat{y}(\mathbf{x}; \mathbf{w}, b) = a(\mathbf{w}\cdot\mathbf{x} + b) = a\left(\sum_{i=1}^{n} w_i x_i + b\right) \tag{4.1}$$

A neuron can be used to model a mapping between the inputs and the output. Without the activation function, a neuron can only be a linear function. If we want to model a non-linear mapping with a neuron, a *non-linearity* must be added into the formulation of neurons. The role of the activation function here is to give the neuron the capability to model non-linear mapping. Historically, the sigmoid function is the standard choice of activation function. Recently, the *rectified linear unit (ReLU)* function has become the default choice since it allows easier training [30].



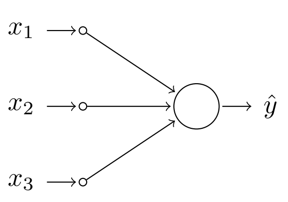
(a) A sigmoid $a(x) = 1/(1 + \exp(-x))$



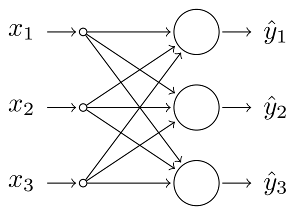
(b) A ReLU $a(x) = max(0, x)$

Figure 2: The two most used activiation functions. The sigmoid function suffers from the vanishing gradient problem for when $|x|$ is large [32]. This leads to slower learning when using gradient descent. ReLu has a constant gradient for $x > 0$ which allowed faster learning [33]. In addition, ReLU is computationally less costly to compute. As a result, ReLU is now the default choice of activation function for ANN [30].

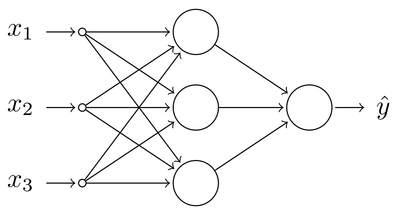
Even with the non-linearity introduced, the set of functions that can be modelled by a neuron is still quite limited. To model a function that has multiple outputs $\mathbf{y} = (y_1, y_2, ..., y_m)$, one can consider a list of n_y neurons. This list of neurons is called a *layer*. To increase the complexity of the model further, one can relay the outputs from a layer to the inputs of another layer. This is then called a *network*. The last layer in a network is called the *output layer*, whereas the other layers are called *hidden layers*. It is proven that given an enough number of neurons in the hidden layer, a network with a single hidden layer can approximate any continuous function up to an error bound [34, 35]. A network is thus called a “universal function approximator”.



(a) A neuron



(b) A layer



(c) A network

Figure 3: Neurons are the most basic building blocks of an ANN. By connecting multiple neurons, the model can be used to approximate more complex functions. Each node represents a neuron and the arrows represents the directions of data flow.

The weights and the biases in each neuron in a network are called *trainable parameters*, or simply parameters. By choosing a suitable network structure and by tuning the trainable parameters correctly, the network can potentially approximate any function. Since an ANN could have many layers and many neurons in each layers, it is very cumbersome to label each of the parameters. In what follows, the symbol ω_j simply represents a parameter and $\boldsymbol{\omega}$ represents the list of all parameters. All the other parameters defining the network are called *hyperparameters*, these include the number of layers, the number of neurons in each layer and the choice of activation function in each neuron, etc. With the exception that the number of neurons in the output layer must matches the desired number of outputs. In other words, the hyperparameters define the set of all possible functions that can be modelled by the network and the trainable parameters define a function in that set of functions. Unfortunately, there are no definitive rules on how to choose the best hyperparameters. Bayesian hyperparameter optimisation is an adaptive method for performing grid search on hyperparameters. Nonetheless, there are many empirical rules of thumb for hyperparameter optimisation [36]. In practice, most of the human effort in the development of an ANN is spent on hyperparameter optimisation. On the other hand, trainable parameters can be optimised by very little human intervention, as detailed in next section.

4.2 How does an ANN learn?

ANN is dedicated to approximate functions that has no analytic expression and often expensive to evaluate. Let $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_y}$ be the function we want to approximate. In order to extract information about the function, we first randomly choose m samples of input $\mathbf{X} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(m)})$ in the domain of the function. The function is then evaluated against the chosen inputs, producing the outputs $\mathbf{Y} = (\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, ..., \mathbf{y}^{(m)}) = (f(\mathbf{x}^{(1)}), f(\mathbf{x}^{(2)}), ..., f(\mathbf{x}^{(m)}))$. These outputs are sometimes called *labels* to distinguish them against the outputs given by the ANN. A 2-tuple $(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})$ is called an *example*. The integer m is then the number of examples and the m examples form a *dataset*.

To begin learning, the hyperparameters are first defined for an ANN. The parameters of the ANN are then initialised randomly. The ANN tries to approach the function f by adjusting the parameters with the information given by the data set. Specifically, the ANN learns by minimising a *loss function* which quantify the differences between the labels $\mathbf{Y}$ in the data set and the outputs given by the ANN $\hat{\mathbf{Y}} = (\hat{\mathbf{y}}^{(1)}, \hat{\mathbf{y}}^{(2)}, ..., \hat{\mathbf{y}}^{(m)})$. A usual choice of loss function is the MS error:

$$\mathcal{L}(\hat{\mathbf{Y}}) = \frac{1}{m} \sum_{i=1}^{m} \left\| \hat{\mathbf{y}}^{(i)} - \mathbf{y}^{(i)} \right\|^2 \tag{4.2}$$

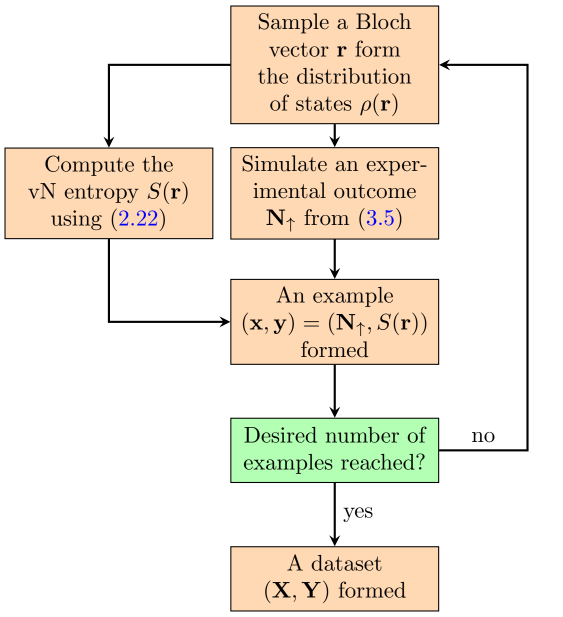
which can be interpreted as the mean squared Euclidean distance between the labels $\mathbf{Y}$ and the outputs given by the ANN $\hat{\mathbf{Y}}$. Notice that the loss function is actually an implicit function $\mathcal{L}(\boldsymbol{\omega})$ of the parameters of the ANN since $\hat{\mathbf{Y}}$ is function of the parameters.

4.3 Minimising the loss function: gradient descent

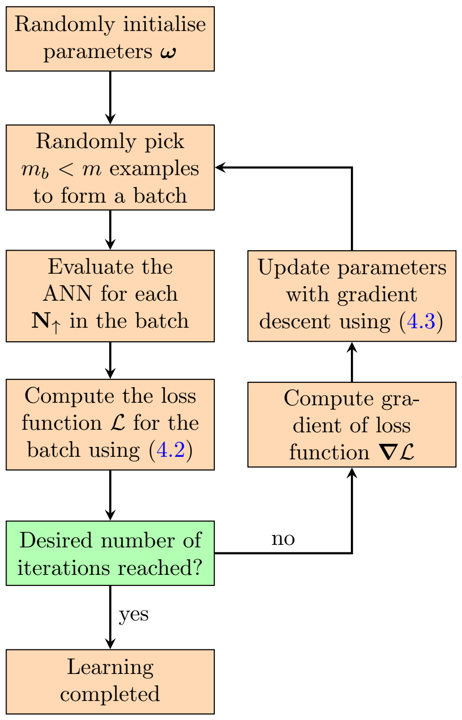
The loss function $\mathcal{L}$ can be minimised, at least in principle, by solving the equation $\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\omega}) = 0$. However, it is often too complicated to be solved algebraically in practice. Instead, numerical optimisation must be deployed. Gradient descent is an iterative numerical optimisation algorithm often used in ANN. The idea of gradient descent is that the negative gradient $-\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\omega})$ points roughly towards the parameters $\boldsymbol{\omega}_{\text{min}}$ that minimises the loss function. A initial point $\boldsymbol{\omega}_0$ is first chosen in the parameter space. We then take a small step along direction of $-\boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\omega})$. The point is updated according to

$$\boldsymbol{\omega}_{n+1} = \boldsymbol{\omega}_n - \alpha_n \boldsymbol{\nabla}\mathcal{L}(\boldsymbol{\omega}_n) \tag{4.3}$$

where α_n is the *learning rate* determines the size of the step. The learning rate is usually chosen adaptively by algorithms such as ADAM [37] and adadelta [38]. When the dataset is too big, it is very costly to compute the loss function $\mathcal{L}$ for every examples. Stochastic gradient descent [39, 40] randomly chooses a subset of the dataset to form a *batch* in each iteration. The evaluation of the loss function is then done for the batch instead of the whole dataset.


Sample a Bloch vector $\mathbf{r}$ form the distribution of states $\rho(\mathbf{r})$
Compute the vN entropy $S(\mathbf{r})$ using (2.22)
Simulate an experimental outcome $\mathbf{N}_\uparrow$ from (3.5)
An example $(\mathbf{x}, \mathbf{y}) = (\mathbf{N}_\uparrow, S(\mathbf{r}))$ formed
Desired number of examples reached?
no
yes
A dataset $(\mathbf{X}, \mathbf{Y})$ formed

(a) The processes of dataset generation. See Appendix C for details on sampling from the distribution $\rho(\mathbf{r})$. Experimental outcomes are simulated by sampling from three independent binomial distributions described in (3.5).


Randomly initialise parameters $\boldsymbol{\omega}$
Randomly pick $m_b < m$ examples to form a batch
Evaluate the ANN for each $\mathbf{N}_\uparrow$ in the batch
Compute the loss function $\mathcal{L}$ for the batch using (4.2)
Desired number of iterations reached?
no
yes
Learning completed
Compute gradient of loss function $\boldsymbol{\nabla}\mathcal{L}$
Update parameters with gradient descent using (4.3)

(b) The processes of the learning algorithm using stochastic gradient descent.

Figure 4: Flow charts summarise the two central algorithms for ANN: the dataset generation and the learning algorithm.

4.4 ANN applied on entanglement estimation

Entanglement estimation is basically a function $f : \{1, 2, ..., N\}^3 \to [0, 1]$. The function takes an experimental outcome $\mathbf{N}_\uparrow$ and gives an estimated entropy $S_{\text{est}} = f(\mathbf{N}_\uparrow)$. The aim of the ANN is to find the best parameters $\boldsymbol{\omega}$ such that the RMS error is minimised. It is important to note that it is not possible to achieve a zero RMS error since the experimental outcome contains insufficient information to uniquely determine the exact entropy. In other words, two Bloch vectors, with distinct vN entropy, can produce the same experimental outcome.

The data generation process is depicted in Figure 4a. In this project, the number of examples in dataset was set to 10^5 and the number of iteration in the learning algorithm was set to 300. The techniques discussed above are all implemented in the python library `tensorflow` [41]. In this project, we used the python library `keras` [42] as a wrapper of `tensorflow` for easier accessibility.

5 Comparison of methods

The aim of entanglement estimation is to give an estimated vN entropy $S_{\text{est}}(\mathbf{N}_\uparrow)$, after given an experimental outcome $\mathbf{N}_\uparrow = (N_{\hat{\mathbf{x}}\uparrow}, N_{\hat{\mathbf{y}}\uparrow}, N_{\hat{\mathbf{z}}\uparrow})$. Here we summarise the three methods discussed in the previous sections:

Scaled direct inversion (SDI):

1. Compute the estimated Bloch vector $\mathbf{r}_{\mathrm{DI}} = 2(\mathbf{N}_\uparrow/N) - 1$.
2. If $\|\mathbf{r}_{\mathrm{DI}}\| > 1$, re-scale it to a unit vector to give $\mathbf{r}_{\mathrm{SDI}} = \mathbf{r}_{\mathrm{DI}}/\|\mathbf{r}_{\mathrm{DI}}\|$.
3. If $\|\mathbf{r}_{\mathrm{DI}}\| \leq 1$, leave it unchanged, i.e. $\mathbf{r}_{\mathrm{SDI}} = \mathbf{r}_{\mathrm{DI}}$.
4. Estimate vN entropy $S_{\mathrm{est}}(\mathbf{N}_\uparrow) = S(\mathbf{r}_{\mathrm{SDI}})$ using (2.22).

Bayesian mean estimation (BME):

1. Find the coefficients B_j in (3.20) to give $\rho(r|\mathbf{N}_\uparrow)$. An algorithm with a time complexity $\mathcal{O}(N^2)$ is detailed in Appendix A.
2. Estimate vN entropy by the mean vN entropy $S_{\mathrm{est}}(\mathbf{N}_\uparrow) = \langle S(\mathbf{r})\rangle_{\mathbf{N}_\uparrow} = \int_0^1 S(r)\rho(r|\mathbf{N}_\uparrow)\,\mathrm{d}r$. This step is carried out with numerical integration.

Artificial neural network (ANN):

1. Generate a dataset using the algorithm in Figure 4a.
2. Train an ANN with the dataset and perform hyperparameter optimisation when necessary.
3. Estimate vN entropy with the output from the trained ANN

5.1 Evaluation of methods

We evaluate the methods by comparing their RMS error. Here we discuss in terms of MS error for notational convenience. There are two ways to compute the MS error $\langle (S_{\mathrm{est}}(\mathbf{r}) - S(\mathbf{r}))^2\rangle_{\mathbf{N}_\uparrow}$:

1. *Semi-analytic:* We have shown how to Semi-analytically compute the MS error $\langle (S_{\mathrm{est}}(\mathbf{N}_\uparrow) - S(\mathbf{r}))^2\rangle_{\mathbf{N}_\uparrow}$ for a particular experimental outcome $\mathbf{N}_\uparrow$ in Section 3.4, where $S_{\mathrm{est}}(\mathbf{N}_\uparrow)$ can be the estimation made by SDI, BME or ANN. The *overall* MS error is given by (3.10):

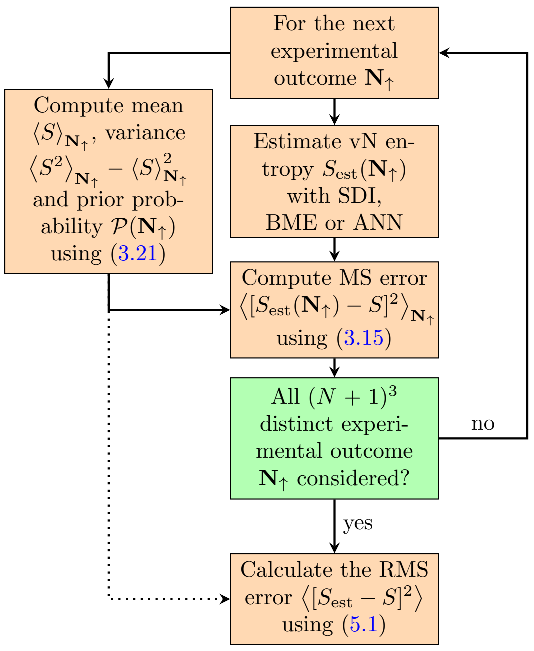
$$\langle (S_{\mathrm{est}}(\mathbf{N}_\uparrow) - S(\mathbf{r}))^2\rangle_{\mathbf{N}_\uparrow} = \langle S^2(\mathbf{r})\rangle_{\mathbf{N}_\uparrow} = \sum_{\mathbf{N}_\uparrow} \mathcal{P}(\mathbf{N}_\uparrow)\,\langle (S_{\mathrm{est}}(\mathbf{N}_\uparrow) - S)^2\rangle_{\mathbf{N}_\uparrow} \tag{5.1}$$

   Notice that (5.1) predicts a *maximum* MS error. It is because the vN entropy $S \in [0,1]$ is bounded from below and above, the residue term $[\langle S(\mathbf{r})\rangle - S_{\mathrm{est}}(\mathbf{N}_\uparrow)]^2$ therefore has a maximum. Notice that the sum has $(N+1)^3$ terms since there are $(N+1)^3$ distinct $\mathbf{N}_\uparrow$ in total. The time complexity for this method is then $\mathcal{O}(N^5)$ since the the computation for the coefficients B_β in (3.20) takes time $\mathcal{O}(N^2)$. Although the symmetries in $\mathcal{P}(\mathbf{N}_\uparrow)$ and thus in $\rho(\mathbf{r}|\mathbf{N}_\uparrow)$ asymptotically reduce the number of terms by a factor of 48 (see Appendix B), the sum is still infeasible to compute when N is large. We thus opt to use numerical methods such as a Monte Carlo method for large N.

2. *Monte Carlo:* First randomly sample m Bloch vectors $(\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, ..., \mathbf{r}^{(m)})$ from the distribution $\rho(\mathbf{r})$. The algorithm for Bloch vector sampling is detailed in Appendix C. Then simulate an experimental outcome ${\mathbf{N}_\uparrow}^{(i)}$ and compute the entropy $S(\mathbf{r}^{(i)})$ for each sampled Bloch vector $\mathbf{r}^{(i)}$. Lastly, compute the approximated MS error

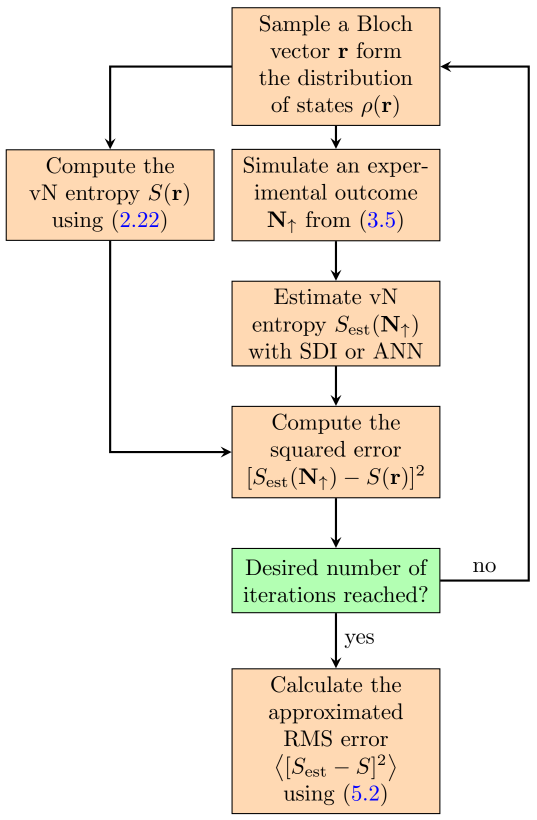
$$\langle (S_{\mathrm{est}}(\mathbf{N}_\uparrow) - S(\mathbf{r}))^2\rangle \approx \sum_i \frac{1}{m}\left[S_{\mathrm{est}}\left({\mathbf{N}_\uparrow}^{(i)}\right) - S\left(\mathbf{r}^{(i)}\right)\right]^2 \tag{5.2}$$

   which should approach the exact MS error when the number of samples m approaches infinity.



For the next experimental outcome $\mathbf{N}_\uparrow$
Compute mean $\langle S\rangle_{\mathbf{N}_\uparrow}$, variance $\langle S^2\rangle_{\mathbf{N}_\uparrow} - \langle S\rangle^2_{\mathbf{N}_\uparrow}$ and prior probability $\mathcal{P}(\mathbf{N}_\uparrow)$ using (3.21)
Estimate vN entropy $S_{\text{est}}(\mathbf{N}_\uparrow)$ with SDI, BME or ANN
Compute MS error $\langle[S_{\text{est}}(\mathbf{N}_\uparrow) - S]^2\rangle_{\mathbf{N}_\uparrow}$ using (3.15)
All $(N+1)^3$ distinct experimental outcome $\mathbf{N}_\uparrow$ considered?
no
yes
Calculate the RMS error $\langle[S_{\text{est}} - S]^2\rangle$ using (5.1)

(a) The semi-analytic method for the calculations of MSE error of SDI, MSE or ANN.



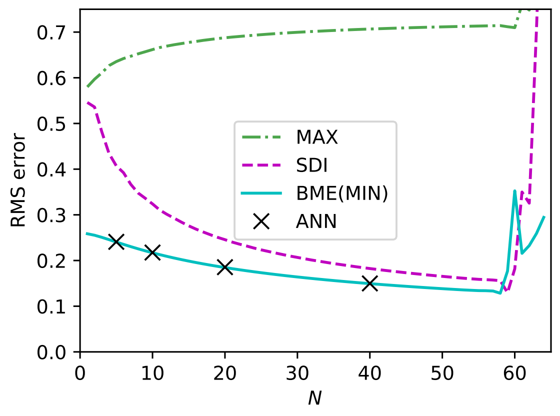
Sample a Bloch vector $\mathbf{r}$ form the distribution of states $\rho(\mathbf{r})$
Compute the vN entropy $S(\mathbf{r})$ using (2.22)
Simulate an experimental outcome $\mathbf{N}_\uparrow$ from (3.5)
Estimate vN entropy $S_{\text{est}}(\mathbf{N}_\uparrow)$ with SDI or ANN
Compute the squared error $[S_{\text{est}}(\mathbf{N}_\uparrow) - S(\mathbf{r})]^2$
Desired number of iterations reached?
no
yes
Calculate the approximated RMS error $\langle[S_{\text{est}} - S]^2\rangle$ using (5.2)

(b) The Monte Carlo method for the calculations of MSE error of SDI or ANN.

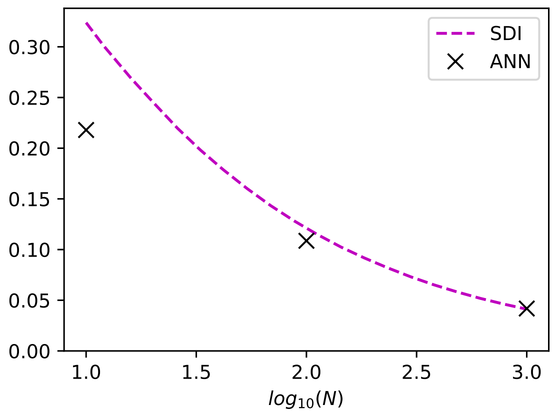
Figure 5: Flow charts summarising algorithms for calculating the MS error.

5.2 Results

In Figure 6, we plotted the RMS errors calculated by the semi-analytic method and the Monte Carlo method. We used the python library `scipy` [43] for our numerics. Since ANN and BME both aims to minimise the MS error, their MS errors should coincide. This fact could be served as a cross-check for our implementation of ANN and BME. The results showed that the difference between the errors produced by ANN and BME ranged from 10^{-4} to 10^{-3}. This small difference was expected since the ANN is an approximation after all. The semi-analytic method showed sign of numerical instability for $N > 57$. We suspect that this due to the accumulated rounding errors in the algorithm. For a large $N \approx 10^3$, the difference between the three methods was shown to be insignificant. However, for small N, BME and ANN were significantly more accurate than SDI.



(a) The RMS error for each methods. ANN was implemented for $N = 5, 10, 20, 40$. The grey out region shows obvious signs of numerical instability For $N > 57$. Notice that the error produced by ANN and BME are too close to be distinguishable in the plot. That means the ANNs are well trained and well optimised.



(b) The RMS error approximated by Monte Carlo simulations. ANN was implemented for $N = 10, 100, 1000$. Note that $\log_{10}(N)$ is plotted instead of N. For $N = 10^3$, SDI did even better than our best trained ANN.

Figure 6: The RMS errors for the entanglement estimation methods. The maximum (MAX) error predicted by (5.1) is also plotted in (a). It can be seen from both (a) and (b) that SDI converges to the BME rapidly. This is expected as the frequencies $N_{\uparrow\mathbf{n}}/N$ tends to the true probabilities $p_{\uparrow\mathbf{n}}$.

6 Conclusion and outlook

We have demonstrated that ANN could be used to estimate the entanglement of pure two-qubit states with projective spin-1/2 measurements. Although BME was shown to minimises the RMS error, our implementation for BME estimator suffered from numerical instability for $N > 57$. In addition, BME may not, and most probably not, be the best estimator under evaluation measures other than RMS error. One could try arbitrary-precision arithmetic to avoid the numerical instability but that would drastically increase the computational cost. We suggest that Monte Carlo integration instead could be use to numerically evaluate the integral defining the mean vN entropy.

On the other hand, ANN was shown to be accurate for pure two-qubit entanglement and requires little human intervention. ANN can be quickly employed to other measurement schemes and evaluation measures. One just need to modify the dataset generation process according to the measurement schemes and switch the loss function used. Both of these tasks are trivial. However, ANN provides no explanation on how the estimations were made and thus give little physical insight. Nonetheless, ANN gives a 'fool-proof' way of finding an upper bound for minimum error achievable. This could be especially useful for more complex systems, such as mixed two-qubit entanglement and 2-qutrit entanglement. ANN can therefore be used to guide analytic research by numerically discovering better estimators than currently known analytic ones.

A relatively easy generalisation of this project is to consider mixed states or two-qutrit states (a pair of 3-level systems). A more challenging extension would be to consider multipartite entanglement. This project only considered for a simple and non-adaptive measurement scheme. An interesting idea is to use reinforcement learning to adaptively select the axis of measurement after each measurement. We believe this could further improve the accuracy obtain in the work.

Acknowledgements

We would like to thank F. Mintert for suggesting this problem for our BSc project. We thank M. I. U. Khalid for the preliminary works done on ANN for the case of $N = \infty$ in his UROP project. Special thanks to R. Mukherjee and F. Suavage for their guidance and useful discussions throughout the project. I appreciate the effort on hyper-parameter optimisation by my project partner.

References

[1] A. Einstein, B. Podolsky, and N. Rosen, Physical Review **47**, 777 (1935).

[2] A. Aspect, P. Grangier, and G. Roger, Physical Review Letters **47**, 460 (1981).

[3] A. Aspect, J. Dalibard, and G. Roger, Physical Review Letters **49**, 1804 (1982).

[4] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, Physical Review Letters **70**, 1895 (1993).

[5] C. H. Bennett and G. Brassard, Theoretical Computer Science **560**, 7 (2014).

[6] A. M. Steane, Physical Review Letters **77**, 793 (1996).

[7] P. W. Shor, Physical Review A **52**, R2493 (1995).

[8] R. P. Feynman, International Journal of Theoretical Physics **21**, 467 (1982).

[9] D. Deutsch, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **400**, 97 (1985).

[10] P. W. Shor, SIAM Journal on Computing **26**, 1484 (1997).

[11] R. Schmied, Journal of Modern Optics **63**, 1744 (2016).

[12] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo, Nature Physics **14**, 447 (2018).

[13] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*, 10th anniversary edition (Cambridge University Press, Cambridge, 2010).

[14] J. von Neumann, *Mathematical Foundations of Quantum Mechanics* (Princeton University Press, 1955).

[15] C. E. Shannon, The Bell System Technical Journal **27**, 379 (1948).

[16] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, Physical Review A **54**, 3824 (1996).

[17] H. P. Robertson, Physical Review **34**, 163 (1929).

[18] W. K. Wootters and W. H. Zurek, Nature **299**, 802 (1982).

[19] M. Paris and J. Rehacek, eds., *Quantum State Estimation* (Springer-Verlag, Berlin, 2004).

[20] R. G. Newton and B.-l. Young, Annals of Physics **49**, 393 (1968).

[21] R. Blume-Kohout and P. Hayden, arXiv:quant-ph/0603116 (2006).

[22] R. Blume-Kohout, New Journal of Physics **12**, 043034 (2010).

[23] D. H. Mahler, L. A. Rozema, A. Darabi, C. Ferrie, R. Blume-Kohout, and A. M. Steinberg, Physical Review Letters **111**, 183601 (2013).

[24] S. Massar and S. Popescu, Physical Review Letters **74**, 1259 (1995).

[25] F. Huszár and N. M. T. Houlsby, Physical Review A **85**, 052120 (2012).

[26] K. S. Kravtsov, S. S. Straupe, I. V. Radchenko, N. M. T. Houlsby, F. Huszár, and S. P. Kulik, Physical Review A **87**, 062122 (2013).

[27] A. Uhlmann, Reports on Mathematical Physics **9**, 273 (1976).

[28] R. Jozsa, Journal of Modern Optics **41**, 2315 (1994).

[29] I. Goodfellow, Y. Bengio, A. Courville, and F. Bach, *Deep Learning* (MIT Press, Cambridge, Massachusetts, 2017).

[30] Y. LeCun, Y. Bengio, and G. Hinton, Nature **521**, 436 (2015).

[31] T. Yiu, *Classification of entanglement using artificial neural networks*, BSc project (2018).

[32] Y. Bengio, P. Simard, and P. Frasconi, IEEE Transactions on Neural Networks **5**, 157 (1994).

[33] V. Nair and G. E. Hinton, in Proceedings of the 27th International Conference on International Conference on Machine Learning, ICML'10 (2010), pp. 807–814.

[34] G. Cybenko, Mathematics of Control, Signals and Systems **2**, 303 (1989).

[35] S. Sonoda and N. Murata, Applied and Computational Harmonic Analysis **43**, 233 (2017).

[36] Y. Bengio, arXiv:1206.5533 (2012).

[37] D. P. Kingma and J. Ba, arXiv:1412.6980 (2014).

[38] M. D. Zeiler, arXiv:1212.5701 (2012).

[39] H. Robbins and S. Monro, The Annals of Mathematical Statistics **22**, 400 (1951).

[40] J. Kiefer and J. Wolfowitz, The Annals of Mathematical Statistics **23**, 462 (1952).

[41] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, in Proceedings of the 12th usenix conference on operating systems design and implementation, OSDI'16 (2016), pp. 265–283.

[42] F. Chollet, *Keras*, `https://github.com/fchollet/keras`, 2015.

[43] E. Jones, T. Oliphant, P. Peterson, et al., *SciPy: open source scientific tools for Python*, 2001.

[44] I. S. Gradshteĭn and D. Zwillinger, *Table of integrals, series, and products*, 8th edition (Elsevier, Amsterdam, 2015).

[45] M. E. Muller, Communications of the ACM **2**, 19 (1959).

[46] M. E. Muller, The Annals of Mathematical Statistics **27**, 569 (1956).

[47] L. J. Bain and M. Englehardt, *Introduction to Probability and Mathematical Statistics* (International Thomson Publishing, 1992).

A Posterior distribution of Bloch length

Here we demonstrate in detail how to evaluate the double integral in (3.19) for the posterior distribution of Bloch length:

$$\rho(r|\mathbf{N}_\uparrow) = \int_{\theta=0}^{\pi} \int_{\phi=0}^{2\pi} \rho(\mathbf{r}|\mathbf{N}_\uparrow) r^2 \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi \tag{A.1}$$

We first write explicitly $\rho(\mathbf{r}|\mathbf{N}_\uparrow)$ in terms of the spherical polar coordinates (r, θ, ϕ). The Bayes' theorem state that

$$\rho(\mathbf{r}|\mathbf{N}_\uparrow) = \frac{\mathcal{P}(\mathbf{N}_\uparrow|\mathbf{r})\rho(\mathbf{r})}{\mathcal{P}(\mathbf{N}_\uparrow)} \tag{A.2}$$

and it is given in (3.5) that

$$\mathcal{P}(\mathbf{N}_\uparrow|\mathbf{r}) = \prod_{\mathbf{n}=\hat{\mathbf{x}},\hat{\mathbf{y}},\hat{\mathbf{z}}} \binom{N}{N_{\uparrow\mathbf{n}}} \left(\frac{1+\mathbf{n}\cdot\mathbf{r}}{2}\right)^{N_{\uparrow\mathbf{n}}} \left(\frac{1-\mathbf{n}\cdot\mathbf{r}}{2}\right)^{N-N_{\uparrow\mathbf{n}}} \tag{A.3}$$

Substituting (A.3) into (A.2) and consider the facts that $\mathcal{P}(\mathbf{N}_\uparrow|\mathbf{r})$ and use that facts that $x = \hat{\mathbf{x}}\cdot\mathbf{r}$, $y = \hat{\mathbf{y}}\cdot\mathbf{r}$ and $z = \hat{\mathbf{z}}\cdot\mathbf{r}$ gives

$$\begin{aligned}\rho(\mathbf{r}|\mathbf{N}_\uparrow) = \eta\,\rho(\mathbf{r})\,&(1+x)^{N_{\uparrow\hat{\mathbf{x}}}}(1-x)^{N-N_{\uparrow\hat{\mathbf{x}}}}\\ &(1+y)^{N_{\uparrow\hat{\mathbf{y}}}}(1-y)^{N-N_{\uparrow\hat{\mathbf{y}}}}\\ &(1+z)^{N_{\uparrow\hat{\mathbf{z}}}}(1-z)^{N-N_{\uparrow\hat{\mathbf{z}}}}\end{aligned} \tag{A.4}$$

where we have defined the normalisation constant

$$\eta = \frac{1}{2^{3N}}\frac{1}{\mathcal{P}(\mathbf{N}_\uparrow)}\binom{N}{N_{\uparrow\hat{\mathbf{x}}}}\binom{N}{N_{\uparrow\hat{\mathbf{y}}}}\binom{N}{N_{\uparrow\hat{\mathbf{z}}}} \tag{A.5}$$

Since we would like to tackle the integrals in spherical polar coordinates, we express $\rho(\mathbf{r}|\mathbf{N}_\uparrow)$ in terms of (r,θ,ϕ):

$$\begin{aligned}\rho(\mathbf{r}|\mathbf{N}_\uparrow) = \eta\,\rho(\mathbf{r})\,&(1+r\sin\theta\cos\phi)^{N_{\uparrow\hat{\mathbf{x}}}}(1-r\sin\theta\cos\phi)^{N-N_{\uparrow\hat{\mathbf{x}}}}\\ &(1+r\sin\theta\sin\phi)^{N_{\uparrow\hat{\mathbf{y}}}}\,(1-r\sin\theta\sin\phi)^{N-N_{\uparrow\hat{\mathbf{y}}}}\\ &(1+r\cos\theta)^{N_{\uparrow\hat{\mathbf{z}}}}\qquad(1-r\cos\theta)^{N-N_{\uparrow\hat{\mathbf{z}}}}\end{aligned} \tag{A.6}$$

Substitute the above into the double integral and use the assumption that the prior is isotropic, i.e. $\rho(\mathbf{r}) = \mathcal{R}(r)$:

$$\rho(r|\mathbf{N}_\uparrow) = \eta\,\mathcal{R}(r)\int_{\theta=0}^{\pi}\left[\int_{\phi=0}^{2\pi} X_+X_-Y_+Y_-\,\mathrm{d}\phi\right]Z_+Z_-r^2\sin\theta\,\mathrm{d}\theta \tag{A.7}$$

where we have defined:

$$X_+ \equiv (1 + r\sin\theta\cos\phi)^{N_{\uparrow\hat{\mathbf{x}}}} = \sum_{i_+=0}^{N_{\uparrow\hat{\mathbf{x}}}} (X_+)_{i_+} (r\sin\theta\cos\phi)^{i_+} \quad \text{where } (X_+)_{i_+} = \binom{N_{\uparrow\hat{\mathbf{x}}}}{i_+} \tag{A.8a}$$

$$X_- \equiv (1 + r\sin\theta\cos\phi)^{N-N_{\uparrow\hat{\mathbf{x}}}} = \sum_{i_-=0}^{N-N_{\uparrow\hat{\mathbf{x}}}} (X_-)_{i_-} (r\sin\theta\cos\phi)^{i_-} \quad \text{where } (X_-)_{i_-} = (-1)^{i_-}\binom{N-N_{\uparrow\hat{\mathbf{x}}}}{i_-} \tag{A.8b}$$

$$Y_+ \equiv (1 + r\sin\theta\sin\phi)^{N_{\uparrow\hat{\mathbf{y}}}} = \sum_{j_+=0}^{N_{\uparrow\hat{\mathbf{y}}}} (Y_+)_{j_+} (r\sin\theta\sin\phi)^{j_+} \quad \text{where } (Y_+)_{j_+} = \binom{N_{\uparrow\hat{\mathbf{y}}}}{j_+} \tag{A.8c}$$

$$Y_- \equiv (1 + r\sin\theta\sin\phi)^{N-N_{\uparrow\hat{\mathbf{y}}}} = \sum_{j_-=0}^{N-N_{\uparrow\hat{\mathbf{y}}}} (Y_-)_{j_-} (r\sin\theta\sin\phi)^{j_-} \quad \text{where } (Y_-)_{j_-} = (-1)^{j_-}\binom{N-N_{\uparrow\hat{\mathbf{y}}}}{j_-} \tag{A.8d}$$

$$Z_+ \equiv (1 + r\cos\theta)^{N_{\uparrow\hat{\mathbf{z}}}} = \sum_{k_+=0}^{N_{\uparrow\hat{\mathbf{z}}}} (Z_-)_{k_-} (r\cos\theta)^{k_+} \quad \text{where } (Z_-)_{k_-} = \binom{N_{\uparrow\hat{\mathbf{z}}}}{k_+} \tag{A.8e}$$

$$Z_- \equiv (1 + r\cos\theta)^{N-N_{\uparrow\hat{\mathbf{z}}}} = \sum_{k_-=0}^{N-N_{\uparrow\hat{\mathbf{z}}}} (Z_+)_{k_+} (r\cos\theta)^{k_-} \quad \text{where } (Z_+)_{k_+} = (-1)^{k_-}\binom{N-N_{\uparrow\hat{\mathbf{z}}}}{k_-} \tag{A.8f}$$

The product $X = X_+X_-$ is then

$$X = X_+X_- = \sum_{i_+=0}^{N_{\uparrow\hat{\mathbf{x}}}} \sum_{i_-=0}^{N-N_{\uparrow\hat{\mathbf{x}}}} (-1)^{i_-}\binom{N-N_{\uparrow\hat{\mathbf{x}}}}{i_-}\binom{N_{\uparrow\hat{\mathbf{x}}}}{i_+}(r\sin\theta\cos\phi)^{i_++i_-} \tag{A.9}$$

The above can be interpret as the polynomial multiplication between X_+ and X_-. Since polynomial multiplication can be evaluated by the convolution the coefficients of the two polynomials, the product $X = X_+X_-$ can also be written as

$$\sum_{i=0}^{N} X_i (r\sin\theta\cos\phi)^i \tag{A.10}$$

where the coefficients X_i of the polynomial X are just the convolution between $(X_+)_{i_+}$ and $(X_-)_{i_-}$:

$$X_i = \sum_{p=0}^{i} (X_+)_p (X_-)_{i-p} \tag{A.11}$$

Similarity, we can obtain the coefficients Y_j and Z_k for the product $Y = Y_+Y_-$ and $Z = Z_+Z_-$ respectively:

$$Y_j = \sum_{p=0}^{j} (Y_+)_p (Y_-)_{j-p} \tag{A.12}$$

$$Z_k = \sum_{p=0}^{k} (Z_+)_p (Z_-)_{k-p} \tag{A.13}$$

Now consider the integrand of the ϕ integral: $XY = X_+X_-Y_+Y_-$:

$$XY = X_+X_-Y_+Y_- = \sum_{i=0}^{N}\sum_{j=0}^{N} X_iY_j(r\sin\theta\cos\phi)^i(r\sin\theta\sin\phi)^j = \sum_{i=0}^{N}\sum_{j=0}^{N} X_iY_j(r\sin\theta)^{i+j}(\cos\phi)^i(\sin\phi)^j \tag{A.14}$$

Substitute the above into the ϕ integral:

$$\int_{\phi=0}^{2\pi} X_+X_-Y_+Y_-\,\mathrm{d}\phi = \sum_{i=0}^{N}\sum_{j=0}^{N} X_iY_j(r\sin\theta)^{i+j}\int_{\phi=0}^{2\pi}(\cos\phi)^i(\sin\phi)^j\,\mathrm{d}\phi \tag{A.15}$$

Now define the integral $T_{ij}^{2\pi} = \int_{\phi=0}^{2\pi}(\cos\phi)^i(\sin\phi)^j\,\mathrm{d}\phi$ for which we will give an recursive formula. The ϕ integral then becomes:

$$\int_{\phi=0}^{2\pi} X_+X_-Y_+Y_-\,\mathrm{d}\phi = \sum_{i=0}^{N}\sum_{j=0}^{N} X_iY_jT_{ij}^{2\pi}(r\sin\theta)^{i+j} \tag{A.16}$$

Putting the above result into $\rho(r|\mathbf{N}_\uparrow)$

$$\rho(r|\mathbf{N}_\uparrow) = \eta\,\mathcal{R}(r)\int_{\theta=0}^{\pi}\left[\sum_{i=0}^{N}\sum_{j=0}^{N} X_iY_jT_{ij}^{2\pi}(r\sin\theta)^{i+j}\right]r^2\sin\theta Z_+Z_-\,\mathrm{d}\theta \tag{A.17}$$

$$= \eta\,\mathcal{R}(r)r\int_{\theta=0}^{\pi}\left[\sum_{i=0}^{N}\sum_{j=0}^{N} X_iY_jT_{ij}^{2\pi}(r\sin\theta)^{i+j+1}\right]Z_+Z_-\,\mathrm{d}\theta \tag{A.18}$$

The polynomial in $(r\sin\theta)$ in the square bracket has many degenerate terms with the same power. We thus need to remove this degeneracy by summing over all the degenerate power:

$$\sum_{i=0}^{N}\sum_{j=0}^{N} X_iY_jT_{ij}^{2\pi}(r\sin\theta)^{i+j+1} = \sum_{\alpha=1}^{2N+1} A_\alpha(r\sin\theta)^\alpha \tag{A.19}$$

where the coefficients A_α are

$$A_\alpha = \sum_{p=0}^{\alpha-1} X_pY_{(\alpha-p-1)}T_{p(\alpha-p-1)}^{2\pi} \tag{A.20}$$

Substitute the above and $Z = Z_+Z_-$ into $\rho(r|\mathbf{N}_\uparrow)$:

$$\rho(r|\mathbf{N}_\uparrow) = \eta\,\mathcal{R}(r)r\int_{\theta=0}^{\pi}\sum_{\alpha=1}^{2N+1} A_\alpha(r\sin\theta)^\alpha\sum_{k=0}^{N} Z_k(r\cos\theta)^k\,\mathrm{d}\theta \tag{A.21}$$

$$= \eta\,\mathcal{R}(r)r\sum_{k=0}^{N}\sum_{\alpha=1}^{2N+1} Z_kA_\alpha r^{k+\alpha}\int_{\theta=0}^{\pi}(\cos\theta)^k(\sin\theta)^\alpha\,\mathrm{d}\theta \tag{A.22}$$

Now define the integral $T_{k\alpha}^{\pi} = \int_{\theta=0}^{\pi}(\cos\theta)^k(\sin\theta)^\alpha\,\mathrm{d}\theta$ for which we will give an recursive formula. The above then becomes

$$\rho(r|\mathbf{N}_\uparrow) = \eta\,\mathcal{R}(r)r\sum_{k=0}^{N}\sum_{\alpha=1}^{2N+1} Z_kA_\alpha T_{k\alpha}^{\pi}r^{\alpha+k} \tag{A.23}$$

$$= \eta\,\mathcal{R}(r)\sum_{k=0}^{N}\sum_{\alpha=1}^{2N+1} Z_kA_\alpha T_{k\alpha}^{\pi}r^{\alpha+k+1} \tag{A.24}$$

Finally, summing over the degeneracy in the powers of the polynomial in r gives

$$\rho(r|\mathbf{N}_\uparrow) = \eta\,\mathcal{R}(r) \sum_{\beta=2}^{3N+2} B_\beta r^\beta \tag{A.25}$$

where the coefficients B_β are

$$B_\beta = \sum_{p=0}^{\beta-1} Z_p A_{(\beta-p-1)} T^\pi_{p(\beta-p-1)} \tag{A.26}$$

The only missing piece is the matrix $T^{2\pi}_{ij}$ and $T^\pi_{k\alpha}$. The two matrices can be constructed with the following recursive formulae [44]

$$\int_0^{n\pi} \sin^p x \cos^q x \,\mathrm{d}x = \frac{p-1}{p+q} \int_0^{n\pi} \sin^{p-2} x \cos^q x \,\mathrm{d}x \tag{A.27}$$

$$= \frac{q-1}{p+q} \int_0^{n\pi} \sin^p x \cos^{q-2} x \,\mathrm{d}x \tag{A.28}$$

where n is an integer.

The constructions of the coefficients array of X_+, X_-, Y_+, Y_-, Z_+, Z_-, and the matrices T^π, $T^{2\pi}$ are trivial and thus not shown here. Here we present an algorithm for the evaluation of the coefficients B_β.

Algorithm 1 Calculate the coefficients B_β. Here, the polynomials are represented by their coefficients array/matrix. It is easy to verify that the following algorithm has a time complexity of $\mathcal{O}(N^2)$.

```
Input: $X_+, X_-, Y_+, Y_-, Z_+, Z_-, T^\pi, T^{2\pi}$
Output: $B$
 1: $X \leftarrow \mathrm{convolve}(X_+, X_-)$;
 2: $Y \leftarrow \mathrm{convolve}(Y_+, Y_-)$;
 3: $Z \leftarrow \mathrm{convolve}(Z_+, Z_-)$;
 4: $XY \leftarrow \mathrm{outer\_product}(X, Y)$;
 5: $XYT \leftarrow (XY) \odot T^{2\pi}$          ▷ symbol $\odot$ refers to element-wise multiplication
 6: for $\alpha = 1$ to $2N+1$ do
 7:     $A_\alpha \leftarrow \sum_{p=0}^{\alpha-1} (XYT)_{p(\alpha-p-1)}$
 8: end for
 9: $ZA \leftarrow \mathrm{outer\_product}(Z, A)$
10: $ZAT \leftarrow (ZA) \odot T^\pi$
11: for $\beta = 1$ to $3N+2$ do
12:     $B_\beta \leftarrow \sum_{p=0}^{\beta-1} (ZAT)_{p(\beta-p-1)}$
13: end for
```

B Symmetries in the probability

The probability $\mathcal{P}(\mathbf{N}_\uparrow|\mathbf{r})$ of getting an experimental outcome $\mathbf{N}_\uparrow$ given an underlying Bloch vector $\mathbf{r}$ is . Note that there two symmetries in this probability. The first symmetry is that the ordering within the experimental outcome does not matter. For instance $\mathcal{P}(\mathbf{N}_\uparrow = (1,2,3)|\mathbf{r}) = \mathcal{P}(\mathbf{N}_\uparrow = (2,1,3)|\mathbf{r}) = \mathcal{P}(\mathbf{N}_\uparrow = (3,2,1)|\mathbf{r})$. This is obvious from (3.5). The second symmetry is the due to a properties of the binomial coefficients:

$$\binom{N}{k} = \binom{N}{N-k} \tag{B.1}$$

One can show that the total number of distinct experimental outcomes, up to the two symmetries is

$$\binom{\lfloor N/2 \rfloor + 3}{3} = \frac{1}{6}(\lfloor N/2 \rfloor + 3)(\lfloor N/2 \rfloor + 2)(\lfloor N/2 \rfloor + 1) \approx \frac{1}{48} N^3 + \mathcal{O}\big(N^2\big) \tag{B.2}$$

The number of distinct experimental outcomes is then reduced by a factor 48 asymptotically.

C Sampling Bloch vectors from isotropic distributions

The training of ANNs and the Monte Carlo method for the error calculates requires sampling Bloch vectors in a distribution $\rho(\mathbf{r})$. Since we are only interested in isotopic distribution of Bloch vector, we first need to generate an isotropic unit vector. This can be done by sampling three real numbers $\mathbf{u} = (u_1, u_2, u_3)$ from a normal distribution. Then obtain the normalised vector $\hat{\mathbf{u}} = \mathbf{u}/\|\mathbf{u}\|$ [45, 46]. We then scale this unit vector by a Bloch length. To sample the Bloch length, we first define the accumulated probability function

$$R(r) = \int_{\|\mathbf{r}\|<r} \rho(\mathbf{r}) \, \mathrm{d}\mathbf{r} \tag{C.1}$$

The probability integral transform [47] tells us that the random variable $r = R^{-1}(x)$ follows the distribution of Bloch length satisfying $\rho(\mathbf{r})$, if $x \in [0,1]$ is uniformly distributed and R^{-1} is the inverse function of R.

For example, consider the volume-uniform distribution $\rho_{\mathrm{vol}}(\mathbf{r}) = 3/4\pi$. The accumulated probability function is then

$$R(r) = \frac{3}{4\pi} \int_{\|\mathbf{r}\|<r} \mathrm{d}\mathbf{r} = \frac{3}{4\pi} \frac{4\pi r^3}{3} = r^3 \tag{C.2}$$

The inverse function is then $R^{-1}(x) = (x)^{(1/3)}$. To sample a Bloch length r, we first sample a $x \in [0,1]$ from a uniform distribution. Then compute $r = R^{-1}(x) = (x)^{(1/3)}$.