Contents

Chapter 1

Introduction

Chapter 2

Multi-armed bandits

2.1 Problem formulation

In the multi-armed bandit problem, there are k distributions ('arms'), $\{P_1, P_2, \ldots, P_k\}$ with unknown means $\{\mu_1, \mu_2, \ldots, \mu_k\}$. For a given number of turns T, the goal is to maximise the expected reward by iteratively selecting a distribution to sample from. In particular, the goal is to minimise the regret defined as

$$R(T) = \sum_{t=1}^{T} \mu^* - \mu_t, \tag{2.1}$$

where μ^* is the mean of the distribution with the highest mean, and μ_t is the mean of the distribution selected at time t. The number of turns T is often referred to as the horizon and can be assumed to be greater than k. This poses a constant struggle between exploration and exploitation, where exploration is the process of trying out new distributions, and exploitation is the process of using the distribution with the highest mean.

Almost always, assumptions are made about the distributions. Otherwise, composing algorithm with any sort of optimality guarantee would be futile. A common assumption, for example, is that the distributions are Bernoulli. Often they are assumed to be Gaussian with unknown mean and maybe some restrictions on the variance. If no assumptions are made to the type of distributions, there are likely to be assumptions made to the variance or support of the distributions.

2.1.1 Best-arm identification

An alternative problem is to find the best arm with as few turns as possible. In this version, a δ is given, and the goal is to find the best arm with probability at least $1 - \delta$.

2.1.2 Bandit generalisations

There are many generalisations to the multi-armed bandit problem. For instance, the distributions may not be stationary, but instead change throughout the game. Alternatively, with contextual bandits, information about a context is given before each turn, which must then be taken into account when selecting an arm. Adversarial bandits complicates matters further, where the rewards are not stochastic from some distribution, but are instead selected by an adversary.

Table 2.1: Comparison of strategies.

Strategy	Regret	Tuning
Random	Linear	NA
Greedy	Linear	NA
Epsilon-greedy	Linear	Difficult
Epsilon-decay	Logarithmic	Difficult
UCB	Logarithmic	Optional
Thompson	Logarithmic	Priors

2.2 Strategies

2.2.1 Explore-only

Pure exploration is obviously a suboptimal strategy, but it is a good baseline to compare against. It can be implemented by selecting an arm uniformly or in order, but it will perform poorly either way. The arm-selection procedure is described by algorithm 2.

Algorithm 1 Random arm selection

```
procedure SelectArm(t)
    Sample i from {1, ..., k} uniformly
    return i
```

It is easy that the expected regret is

$$R(T) = T\left(\mu^* - \frac{1}{k}\sum_{i=1}^{k}\mu_i\right), \tag{2.2}$$

which is $\Theta(T)$. This motivates the search for an algorithm with sublinear regret.

2.2.2 Greedy

A simple algorithm and a good baseline is the greedy algorithm. Here, all arms are tried N initial times, and the empirical means are used to select the best arm. Afterwards, the arm with the highest empirical mean is selected for all remaining turns. The arm-selection procedure is listed in algorithm 2, where $\hat{\mu}_i$ is the empirical mean of arm i.

Algorithm 2 Greedy arm selection

```
procedure SelectArm(t)
    if t ≤ Nk then
        return (t mod k) + 1
    else
        return argmax_{i=1,...,k} μ̂_i
```

With greedy selection, the expected regret is

$$R(T) = \sum_{t=1}^{T} \mu^* - \hat{\mu}_t, \tag{2.3}$$

2.2.3 Epsilon-greedy

The problem with the greedy algorithm is that it may be unlucky and not discover the best arm in the initial exploration phase. To mitigate this, the epsilon-greedy algorithm may be used. In this algorithm, the estimated arm is pulled with probability $1 - \epsilon$ and a random arm is pulled with probability ϵ. This ensures convergence to correct exploitation as the horizon increases, and it will generally reduce the regret. Still, with a constant ϵ, a constant proportion of the turns will be spent exploring, keeping the regret linear in the horizon. Choosing ϵ is a trade-off between exploration and exploitation and can significantly affect the regret.

Epsilon-decay

If one allows the ϵ to decay over time, the regret can be reduced even further. This makes sense, as exploration is less worthwhile as estimates become ever more accurate. A common choice is to decay ϵ as $\epsilon_t = \epsilon_0/t$.

2.2.4 UCB

The upper confidence bound (UCB) algorithm is a more sophisticated algorithm based on estimating an upper bound for the mean of each arm. One always chooses the arm whose upper confidence bound is highest, a principle known as 'optimism in the face of uncertainty'.

In the original formulation, where support on only $[0, 1]$ is assumed, the upper confidence bound is given by

$$\mathrm{UCB}_t(a) = \hat{\mu}_t(a) + \sqrt{\frac{2 \ln t}{N_t(a)}}, \tag{2.4}$$

where $\hat{\mu}_t(a)$ is the empirical mean of arm a at time t, $N_t(a)$ is the number of times arm a has been pulled at time t. It generally achieves logarithmic regret.

Many variants of the UCB algorithm exist. Different assumptions about the distributions changes the confidence bounds.

2.2.5 Bayesian: Thompson sampling

Thompson sampling is a Bayesian approach to the multi-armed bandit problem, originally described in 1933 as a way to handle the exploration-exploitation dilemma in the context of medical trials. The idea is to sample from the posterior distribution of the means of the arms and pull the arm with the highest sample.

2.3 Simulations

Chapter 3

Quantum computing

The field of quantum computing is split into two main branches: the development of quantum hardware and the study of algorithms that use such hardware. Only the second branch is relevant for this thesis, and even so only a brief explanation is offered here. For more details, see [**nielsen2012**] for a rigorous, complete description or [**qiskit_textbook**] for an introduction focused on programming. Any reader should have a basic understanding of linear algebra and classical computing. Knowledge of quantum mechanics is not assumed, albeit certainly helpful.

3.1 Quantum states

3.1.1 The qubit

The quantum bit, the qubit, is the building block of quantum computing. Like the classical binary digit, it can be either 0 or 1. But being quantum, these are quantum states, $|0\rangle$ and $|1\rangle$[1], and the qubit can be in any superposition of these states. This follows from the first postulate of quantum mechanics[2], which states that an isolated system is entirely described by a normalised vector in a Hilbert space. For the qubit, this is the two-dimensional space where the states $|0\rangle$ and $|1\rangle$ are basis vectors, known as the computational basis states. Thus, the state of a qubit can be expressed as

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}, \tag{3.1}$$

where $\alpha, \beta \in \mathbb{C}$. The only requirement is that the state is normalised, $|\alpha|^2+|\beta|^2 = 1$. Normalisation is required due to the Born rule, as the absolute square of the coefficients is the probability of measuring the qubit in the corresponding basis state.

3.1.2 The Bloch sphere

A useful tool for visualising the state of a qubit is the Bloch sphere. First, it should be noted that states on the form eq. (3.1) are not physically unique, only the relative complex phase matters. There is a global phase which can not be observed, and so it is not physically relevant. Taking that and the normalisation into account, the state of the qubit can be expressed as

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle, \tag{3.2}$$

where $\theta, \phi \in \mathbb{R}$. Interpreting θ as the polar angle and ϕ the azimuthal angle, the state of the qubit can be identified with a point on a sphere. See fig. 3.1. The state $|0\rangle$ is typically thought of as the north pole of this sphere and $|1\rangle$ the south pole.

[1] The $|\cdot\rangle$ notation is known as a ket and is used in quantum mechanics to denote a quantum state. It is effectively a column vector. The inner product may be taken with a bra, $\langle\cdot|$, to give a scalar. These inner products are then denoted by $\langle\cdot|\cdot\rangle$. Similarly, outer products are well-defined and denoted by $|\cdot\rangle\langle\cdot|$.

[2] As they are laid out in **[nielsen2012]**.

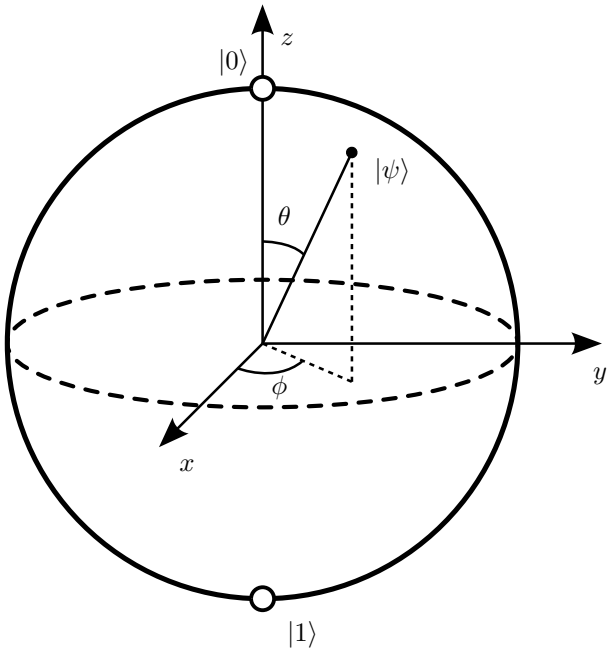


Figure 3.1: The Bloch sphere. On it, the state of a single qubit state is represented by a point. The state $|0\rangle$ is the north pole, and $|1\rangle$ is the south pole. The latitudinal angle θ determines the probability of measuring the qubit in the state $|0\rangle$, while the longitudinal angle ϕ corresponds to the complex phase between the two basis states. From **[wikipedia_bloch]**.

3.1.3 Mixed states and density operators

It is not only the superpositions of states that are important in quantum computing, but also the mixed states, states that are statistical ensembles of pure states. Pure states are those expressible as a single ket like eq. (3.1), while mixed states arise when the preparation the system is not perfectly known or when the system interacts with the environment. For the description of mixed states, the formalism of density operators is more useful than the state vector formalism. If there are no classical uncertainties, the state is pure, and the density operator can be expressed a single ket-bra, $\rho = |\psi\rangle\langle\psi|$. In a mixed state, however, some classical probabilities p_i are associated with the different pure states $|\psi_i\rangle$, and the state of the system is described by the density operator

$$\rho = \sum_{i=1}^{n} p_i \, |\psi_i\rangle\langle\psi_i| \tag{3.3}$$

where $|\psi_i\rangle$ are the states of the system, and $\langle\psi_i|$ are the corresponding dual vectors. Being probabilities, the p_i must be non-negative and sum to one. Given a basis and a finite Hilbert space, the density operator can be expressed as a density matrix[3] where the diagonal elements are the probabilities of measuring the system in the corresponding basis state. Furthermore, it is easily seen that the density operator must be positive semidefinite and Hermitian.

The Pauli matrices,

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{3.4}$$

together with the identity matrix serve as a basis for the real vector-space of Hermitian 2×2 matrices. Since the diagonal elements of a density matrix must sum to one, the density matrix for a single qubit can be expressed as

$$\rho = \frac{1}{2} \left(I + x\sigma_x + y\sigma_y + z\sigma_z \right), \tag{3.5}$$

where $x, y, z \in \mathbb{R}$. Being positive semidefinite, the determinant should be non-negative, and thus it can be shown that $x^2 + y^2 + z^2 \leq 1$. This allows density operators to be interpreted as points on the Bloch sphere or indeed within it. Notably, pure states lie on the surface, while mixed states lie within the sphere (or rather, the Bloch ball). A pure quantum superposition of $|0\rangle$ and $|1\rangle$ with equal probabilities would have a complex phase and lie somewhere on the equator, while a statistical mixture with equal classical probabilities of being $|0\rangle$ and $|1\rangle$ would lie in its centre.

[3] Density operators and matrices are often used interchangeably in quantum computing. Due to the finite number of qubits, the Hilbert spaces are always finite-dimensional, and with the canonical basis, there is a canonical way of representing density operators as matrices.

3.1.4 Systems of multiple qubits

Although the continuous nature of the qubit is indeed useful, the true power of quantum computers lie in how multiple qubits interact. Having multiple qubits enables entanglement, which is a key feature of quantum computing.

With two qubits, there are four possible states, $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. Each of these four states have their own probability amplitude, and thus their own probability of being measured. A two-qubit system will therefore operate with four complex numbers in the four-dimensional Hilbert space $\mathbb{C}^4$.

Generally, the state of multiple qubits can be expressed using the tensor product as

$$|\psi_1\psi_2\cdots\psi_n\rangle = |\psi_1\rangle\,|\psi_2\rangle\cdots|\psi_n\rangle = |\psi_1\rangle\otimes|\psi_2\rangle\otimes\cdots\otimes|\psi_n\rangle\,. \tag{3.6}$$

What makes this so powerful is that the state of a multi-qubit system has the general form

$$\begin{aligned} |\psi_1\psi_2\cdots\psi_n\rangle &= c_1\,|0\ldots00\rangle + c_2\,|0\ldots01\rangle + \cdots + c_{2^n}\,|1\ldots11\rangle \\ &= (c_1, c_2, \ldots, c_{2^n})^\top \in \mathbb{C}^{2^n}, \end{aligned} \tag{3.7}$$

which means that with n qubits, the system can be in any superposition of the 2^n basis states. Operating on several qubits then, one can do linear algebra in an exponentially large space. This is a key part of the exponential speed-ups possible with quantum computers.

3.2 Quantum operations

3.2.1 Single-qubit gates

To operate on one or more qubits, a unitary operation is applied to the state. This is a computational interpretation of the unitary time evolution resulting from a Hamiltonian acting on the (closed) quantum system, described by the second postulate of quantum mechanics and the Schrödinger equation. As the operations are unitary, a pure state remains pure. These operations are often thought of as gates, paralleling the classical gates in digital logic. Mathematically, with a finite number of qubits, a unitary gate U can be expressed as matrices acting on the state vector, $|\psi\rangle$, as

$$|\psi'\rangle = U\,|\psi\rangle\,, \tag{3.8}$$

where $|\psi'\rangle$ is the resulting state.

The most basic gates are the Pauli gates, which are applications of the Pauli matrices from eq. (3.4) and are as gates simply denoted as X, Y, and Z. These gates can be seen as half turns around the x-, y- and z-axes, respectively, of the Bloch sphere. The X-gate is also known as the NOT gate, as it mirrors the classical NOT gate by mapping $|0\rangle$ to $|1\rangle$ and vice versa. It is however more general, being also applicable to superposition states.

The Hadamard gate,

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \tag{3.9}$$

is a rotation around the line between the x- and z-axes by $\pi/2$. It is an important gate in quantum computing, as it is used to create superpositions of the computational basis states. Applied on an initial $|0\rangle$ state, it creates the entangled state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. Two consecutive applications thereof returns the state to the initial state, as can be seen from the matrix squaring to the identity.

The R_X-, R_Y- and R_Z-gates are rotations around the x-, y- and z-axes, respectively, by an arbitrary angle θ:

$$R_X(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix},$$
$$R_Y(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix},$$
$$R_Z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}.$$

These parametrised gates will be useful in **??**.

3.2.2 Multi-qubit gates

Multi-qubit gates are gates that act non-trivially on more than one qubit. The most used multi-qubit gate is the controlled X-gate, also known as the CNOT. Being controlled means that it only acts on the second qubit if the first qubit is in the state $|1\rangle$. Of course, the first qubit may be in a superposition, and the CNOT this way allows for the creation of entanglement between the two qubits. If the first qubit has probability amplitude α of being in the state $|1\rangle$, the second qubit will have probability amplitude α of being flipped. The CNOT-gate, acting on the leftmost qubit in the tensored two-qubit system can be expressed in matrix form as

$$\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{3.10}$$

In theory, any unitary single-qubit operation can be controlled. However, it is often only the CNOT that is used is implemented in the hardware. Another interesting two-qubit gate is the controlled Z-gate, CZ, expressible as the matrix

$$\mathrm{CZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \tag{3.11}$$

Because it only alters the amplitude of $|11\rangle$, it does not actually matter which qubit is the control and which is the target.

3.2.3 Observables and measurements

For an output to be obtained from a quantum computer, a measurement must be performed. This is typically done at the end of all operations and of all qubits, where each qubit is measured in the computational basis to yield a string of bits.

As described by the third postulate of quantum mechanics, any observable quantity has a corresponding Hermitian operator A, spectrally decomposable as $A = \sum_i \lambda_i P_i$, where λ_i are the (necessarily real) eigenvalues and P_i are the corresponding projectors onto the eigenspaces. When measuring, the probability of obtaining the outcome λ_i is given by

$$p_i = \langle\psi| P_i |\psi\rangle, \tag{3.12}$$

where $|\psi\rangle$ is the state before the measurement. It is one of Nature's great mysteries what exactly a measurement is and even more so how and why it is different from the unitary evolution described by the second postulate. In the quantum

computational setting, it can be thought of as taking a random sample with the probabilities as given by the above equation.

Often, the underlying probabilities are what is of interest. Therefore, many measurements will be performed. Usually, these results are averaged to obtain an estimate, but more complicated post-processing methods are also possible. For instance, neural networks have shown useful properties in regard of reducing variance in the estimates, though at the cost of some bias [**torlai2020**].

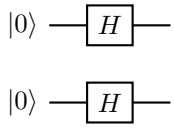
Canonically, the computational Z-basis is used for measurements, and it is usually the only basis for which measurements are physically implemented in a quantum computer. When measuring in the computational basis in which a state is expressed, as eq. (3.7), the probabilities are simply given by the absolute square of the coefficients. To virtually measure another observable, a change of basis is performed. This is achieved by applying a unitary transformation before measurement.

Measurements may be done in the middle of a computation and be used to control gates. If the qubits are entangled, measuring one will affect the measurement probabilities of others. Using such intermediate measurements is a way of introducing non-linearities in the otherwise unitary nature of the unmeasured quantum world.

3.2.4 Quantum circuits

The operations on qubits are often described using quantum circuits, which are a graphical representation of the operations on the qubits, the quantum algorithms. They are read from left to right. It is standard procedure to assume all qubits start in the state $|0\rangle$. Gates are generally written as boxes with the name of the gate inside.

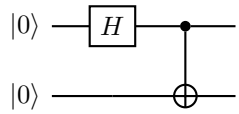
A simple example is the circuit



, (3.13)

which prepares the state $\frac{1}{2}(|00\rangle+|01\rangle+|10\rangle+|11\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)\otimes\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. This is a pure state with no entanglement, and so the measurement probabilities of the two qubits are independent. When measured, all four outcomes are equally likely.

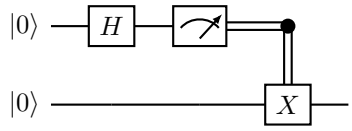
Slightly more interesting is the circuit



(3.14)

in which the first qubit is put into a superposition using the Hadamard gate before a CNOT gate is applied to the second, controlled by the first. This creates the state $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. The measurement probabilities of the two qubits are now correlated; if the first qubit is measured to be $|1\rangle$, the second will always be $|1\rangle$ and vice versa. The probability of measuring the qubits to be different is nil.

To create a mixed state, an intermediate measurement can be used to control a gate. For instance, the circuit



(3.15)

places the second qubit in the mixed state $\frac{1}{2}(|0\rangle\langle 0| + |1\rangle\langle 1|)$. If it were immediately to be measured, it would have a 50% chance of being $|0\rangle$ and a 50% chance of being $|1\rangle$. The uncertainty is only classical, and it could therefore not be used to create entanglement or for any other quantum spookiness.

3.2.5 Quantum supremacy

Exponential speed-ups do not come for free. Although the states spaces are exponentially large, with only a limited set of operations available, states can not be created and manipulated arbitrarily; the problem must have some structure to be exploited for a speed-up to be possible. Quantum computers do only solve certain problems more efficiently than classical computers, and finding the algorithms to do so is not trivial. Shor's algorithm has time complexity $O((\log N)^3)$ while the most efficient known classical algorithm, the general number field sieve, is sub-exponential with a time complexity on the form $\Omega(k^{\frac{1}{3}} \log^{2/3} k)$, where $k = O(2^N)$ [**dervovic2018**]. To solve linear system, there is the HHL algorithm with time complexity $O(\log(N)\kappa^2)$, where κ is the condition number. This is an exponential speed-up over the fastest known classical algorithm[4], which has time complexity $O(N\kappa)$. Still, these are non-trivial algorithms, not yet usable in practice and that were not easily found.

[4]Given that the condition number does not grow exponentially. There are also difficulties in loading the data into the quantum computer and extracting the solution that could negate any exponential speed-up. C.f. [**aaronson2015**].

Polynomial speed-ups are perhaps more easily found. For example, the Grover algorithm which is used to search for an element in an unsorted list has time complexity $O(\sqrt{N})$ [**grover1996**]. Classically, this can not be done in less than $O(N)$ time. It can be proven that the Grover algorithm is optimal [**zalka1999**], so for this problem, an exponential speed-up is impossible. This algorithm and the more general amplitude amplification on which it builds solves very general problems and are often used subroutines to achieve quadratic speed-ups in other algorithms. Being only a quadratic speed-up, it is not as impressive as the exponential speed-ups, and achieving quantum supremacy in that manner would require larger quantum computers than if the speed-up were exponential.

It is proven that the class of problems quantum computers can solve in polynomial time (with high probability), BQP, contains the complexity class P [**nielsen2012**]. This follows from the fact than quantum computers run do any classical algorithm. Since quantum computers can solve problems like integer factorisation and discrete logarithms efficiently, it is believed that BQP is strictly greater than P, but as whether P = NP remains unknown, these problems could actually be in P. In a similar vein, NP-complete problems are believed to lie outside BQP.

3.3 Quantum algorithms

3.3.1 Grover's algorithm

The quantum search algorithm of Grover [**grover1996**] is a quantum algorithm that finds an element in an unstructured list with high probability. While such a problem necessarily requires $O(N)$ time in a classical setting, needing on average $N/2$ steps to find the element and in the worst case N, Grover's algorithm finds the element in $O(\sqrt{N})$ steps. This is a quadratic speed-up.

Grover's algorithm is provably optimal; no quantum algorithm can perform such general searches faster. This should not be surprising. If an exponential speed-up were possible, it could be used to find the solution to NP-hard problems fast.

For Grover's algorithm to work, assume there is a function $f : \{0, \ldots, N-1\} \to \{0, 1\}$ that maps the index of an element to 1 if it is the one desired and 0 otherwise. Then, one assumes access to a quantum oracle, $\mathcal{O}_f$ (effectively a black box subroutine) that implements f thus:

$$\mathcal{O}_f \left|x\right\rangle = (-1)^{f(x)} \left|x\right\rangle . \tag{3.16}$$

A single application of this oracle is not enough to find the desired element, as the square of the amplitude of the desired element remains unchanged. Central to Grover's algorithm is the idea of amplifying the amplitude of the desired element. This is done by applying a sequence of operations that is repeated until the amplitude of the desired element is large enough for it is most likely to be measured, while the amplitudes of the other elements are reduced.

Let the state $\left|w\right\rangle$ which be the winner state, a state with amplitude 1 for the desired element and 0 for all others. Then consider the state $\left|s\right\rangle$, which is a uniform superposition state, a state with equal amplitudes for all elements. Define the state $\left|s'\right\rangle$ by subtracting the projection of $\left|w\right\rangle$ onto $\left|s\right\rangle$ from $\left|s\right\rangle$:

$$\left|s'\right\rangle = \left|s\right\rangle - \left\langle w|s\right\rangle \left|w\right\rangle . \tag{3.17}$$

These two orthogonal states from a basis of a two-dimensional subspace of the greater Hilbert space. The uniform superposition state $\left|s\right\rangle$ serves as a starting point for the algorithm, and is achieved by applying Hadamard gates to all qubits. It is expressible as $\left|s\right\rangle = \cos(\theta) \left|s'\right\rangle + \sin(\theta) \left|w\right\rangle$, where $\theta = \arcsin \left\langle s|w\right\rangle = \arcsin(1/\sqrt{N})$.

Applying the oracle on $\left|s\right\rangle$ leaves its $\left|s'\right\rangle$ component unchanged, but flips the sign of the $\left|w\right\rangle$ component. This results in the state $\left|\psi\right\rangle = \cos(-\theta) \left|s'\right\rangle + \sin(-\theta) \left|s'\right\rangle$, which can be seen as reflection of $\left|s\right\rangle$ in the $\left|s'\right\rangle$ direction.

Next, the state $\left|\psi\right\rangle$ is reflected about the initial $\left|s\right\rangle$ state, resulting in the state $\left|\psi'\right\rangle = \cos(3\theta) \left|s'\right\rangle + \sin(3\theta) \left|w\right\rangle$. Reflection thus is achieved by the diffusion

operator $R = H^{\otimes n} S_0 (H^{\otimes n})^{-1} = H^{\otimes n} S_0 H^{\otimes n}$, where $S_0 = 2\,|0\rangle\langle 0| - I$ is the reflection operator about the $|0\rangle$ state, that is an operator that flips the sign of all but the $|0\rangle$ component.

The product of the oracle and the diffusion operator defines the Grover operator, which is simply applied until the amplitude of the $|w\rangle$ is sufficiently amplified. After k iterations, the state is $|\psi_k\rangle = \cos((2k+1)\theta)\,|s'\rangle + \sin((2k+1)\theta)\,|w\rangle$. Measuring the correct state has probability $\sin^2((2k+1)\theta)$. Therefore, $k \approx \pi/4\theta$ iterations should be completed. Assuming large N, for a short list would not warrant the use of Grover's algorithm, $\theta = \arcsin(1/\sqrt{N}) \approx 1/\sqrt{N}$, and so $k \approx \pi\sqrt{N}/4$.

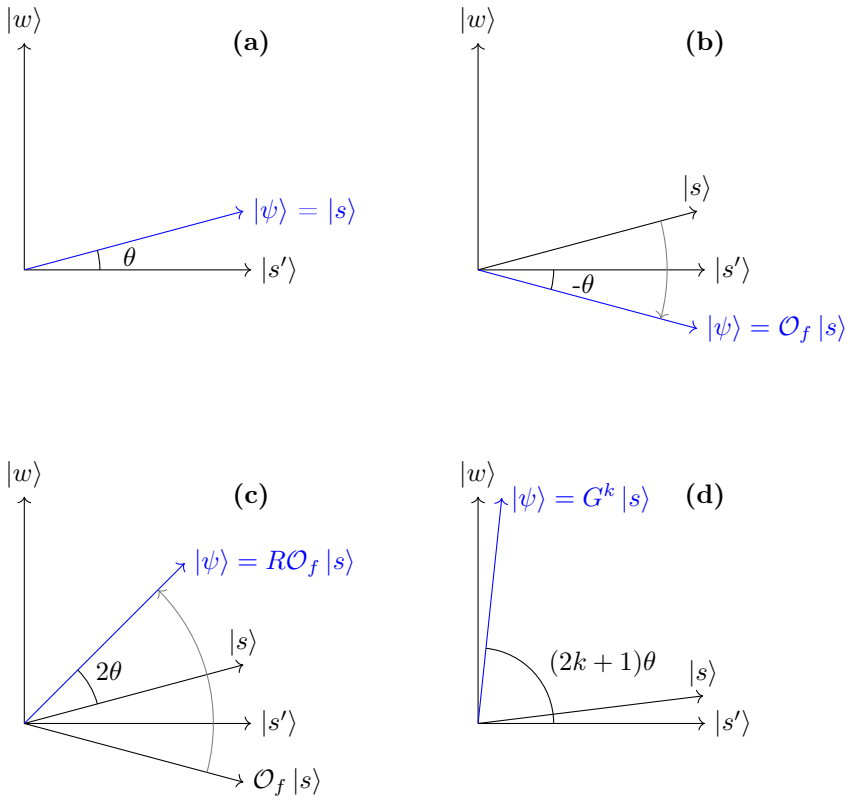


Figure 3.2: Grover's algorithm visualised. (a) The initial uniform superposition state $|s\rangle$ is prepared, which can be seen as a linear combination of $|w\rangle$ and $|s'\rangle$, forming an angle θ to the s'-axis. (b) The oracle $\mathcal{O}_f$ is applied to $|s\rangle$, flipping the sign of its $|w\rangle$ component, inverting the angle θ. (c) The reflection operator R is applied, reflecting the state about the initial state and towards the goal, resulting in a state with an angle 3θ to the w-axis. (d) After repeating the previous two steps a k times, the angle is $2k + 1\theta$, and if k is chosen wisely, this means that the system is in a state close to the desired state $|w\rangle$, such that measuring the system will likely result in $|w\rangle$.

Chapter 4

Quantum bandits

Several formulations of the multi-armed bandit problem have been made for a quantum computing setting. As the central issue in bandit problems lie in sample efficiency rather than computational difficulties, quantum computers offer little advantage assuming classical bandits. However, by allowing bandits to be queried in superposition, major speed-ups can be achieved. For such bandits, regret minimisation is no longer a valid objective, and instead the problem is to find a strategy that maximises the probability of finding the optimal arm with as few queries as possible.

4.1 Casalé

In [**casale2020**], an algorithm based on amplitude amplification is proposed and is shown to find the optimal arm with quadratically fewer queries than the best classical algorithm for classical bandits — albeit with a significant drawback: the probability of the correct arm being suggested can not be set arbitrarily high, but is instead given by the ratio of the best arm's mean to the sum of the means of all arms. This