Master Thesis:
Machine Learning for Nonlocality Detection in Multipartite System

Université Claude Bernard, Villeurbanne (Lyon), France

ICFO-Institut de Ciènces Fotoniques, Castelldefels (Barcelona), Spain

Supervisors:

Joseph Bowles, Gabriel Senno, and Antonio Acin

Xavier Valcarce

1th August 2018

Abstract

Nonlocality, one of the most intriguing aspects of quantum theory, is the key resource in many of quantum mechanics' advantages for information processing tasks. Therefore, developing protocols to efficiently certify the nonlocal nature of some observed multipartite correlations has become a crucial endeavour in quantum information theory. In most of the studied nonlocality scenarios, deciding whether a set of correlations is nonlocal or not amounts to solving a linear program, for which efficient algorithms exist. However, the size of those linear programs is exponential in the number of particles in the many-body system under study, becoming infeasible even for small sized systems.

In this thesis, we combine state-of-the-art machine learning algorithms and recent advancements in the theoretical study of locality in multipartite systems to develop an efficient and highly accurate protocol for certifying quantum nonlocality. Our learning algorithms are trained with sample correlations from small sized systems and then tested with correlations from systems with increasing number of parties (where running the linear program becomes infeasible). Through the use of domain adaptation techniques, we are able to achieve instantaneous detection of nonlocal correlations with more than 60% accuracy and almost independently of the number of parties.

Contents

Chapter 1

Introduction

One of quantum theory's features which have puzzled scientists the most since its origin is nonlocality, the fact that measuring a property of a quantum system can instantaneously determine the results of another property measured on a distant system. Such kind of nonlocal influence was part of an important debate inside the scientific community. In their article of 1935 entitled "Can quantum-mechanical description of physical reality be considered complete?", Einstein, Podolsky and Rosen [1] argued that any theory making the same predictions as quantum theory and, at the same time, avoiding such *spooky action at a distance*, as they called these non-local influences, has to postulate the existence of "real properties" (or, *hidden variables*) which, when taken into account, allow for the complete local determination of the observations' outcomes. Since orthodox quantum theory does not include these, from the assumption of the impossibility of nonlocal causation one has to conclude its incompleteness. Decades later, in 1964, John S. Bell proved that the predictions of quantum mechanics can never be explained by a physical theory of *local hidden variables*, under the assumption of *free will*, going against EPR's intuition [2].

Besides producing a fundamental change in our perception of the universe, the study of Bell nonlocality [3] has led to new technological applications, and now we know that nonlocal correlations are the key resource in most of quantum mechanics' advantages for informational and computational tasks; key distribution protocols [4], algorithms for distributed computation [5], or random number generators [6, 7] are exemples of such applications [8]. Certifying the nonlocality of the statistics of measurements over multipartite quantum systems has therefore become a key problem in quantum information theory. This certification can be done numerically using

convex optimization algorithms to solve a membership problem. However, given that the dimension of the space of correlations grows exponentially with the number of quantum systems, characterizing (non)local correlations of a multipartite system is, in general, of time complexity exponential in the number of parties [9, 10].

Machine learning [11], the core of artificial intelligence, is a fast-expanding interedisceplinary field. Recently, it has been used in nearly all fields of science, including quantum many-body systems [12, 13, 14]. In a nutshell, the the goal here is to develop efficient (albeit not perfectly accurate) schemes for different classification tasks by training learning algorithms on large datasets (possibly with the aid of, albeit inefficient, perfect classification algorithms) aiming at making the error in generalizing the classification to unseen data as small as possible.

In this thesis, we set out to develop an efficient and highly accurate scheme to detect nonlocality in many-body systems. In the spirit of Machine Learning, our goal is to train a learning algorithm on correlations from systems with a small number of parties with the aim that the classification scheme generalizes well to a larger number of parties. This already posses a technical problem, as it implies that the dimensions of the space of training and testing points differ. We deal with this by working in the space of symmetrized correlators, introduced by Tura et al. in [15], whose dimension is independent of the number of parties N of the multipartite system under study. Although looking only at symmetrized correlators implies loosing information, this restricted space is already sufficient for certyfying nonlocality in many experimentally relevant settings, e.g. for ground states of the two-body Lipkin-Meshkov-Glick Hamiltonian [15]. However allowing us to work in a constant dimensional space, the shape of the set of local correlations within this space will still vary with N and, to cope with this second issue, we will use a recently-developed ML scheme called *Domain Adversarial Neural Network* [16] whose aim is to generate classifiers which "ignore" certain features of the test data (in our setting, the number of parties N).

This thesis is organized as follows. In Chapter 2 we give an introduction to Bell nonlocality and to the setting of symmetrized correlators introduced in [15]. Next, in Chapter 3, we describe the Machine Learning algorithms we used, explain how we generated the training and test data and report the results of the classification scheme developed. Finally, in Chapter 4 we present our conclusions and discuss future research directions.

Chapter 2

Preliminaries – Locality and Convex Geometry

In this chapter, the concept of locality is introduced and some specific aspects are detailed. First, nonlocality is defined with the use of a Bell game. Then, it is shown how one can represent and analyse locality using convex geometry. These two topics are also discussed in the context of multipartite systems. Finally, a recent improvement in the study of nonlocality in multipartite systems, through the use of symmetric n-body correlators, is reviewed.

2.1 Characterizing Locality via a Bell Game

Local correlations – A typical Bell game consists of two players (or, *parties*), Alice and Bob, who may have been in contact in the past, and are now spatially separated and without the ability to communicate with each other[1]. Each player has a physical system on which he/she can perform, in each round of the game, one of a finite number of measurements (or, inputs) each of them having a finite number of results (or, outputs). We label Alice's (resp. Bob's) input as x (resp. y) and the corresponding output as a (resp. b). The object of interest is $p(ab|xy)$: the probability that when performing measurements x and y the players get outcomes a and b (see Fig. 2.1). We will arrange these probabilities in a vector $\mathbf{P} := \{p(ab|xy)\} \in \mathbb{R}^{d_A d_B m_A m_B}$, where m_A (resp. m_B) is the number of measurement Alice (resp. Bob) can perform, and d_A (resp. d_B) corresponds to the number of outcomes of those

[1] Ideally separated by a space-like interval, this to ensure the non-signaling constraint (cf. 2.1 **non-signaling correlation**)

measurements (considering that all measurements have the same number of possible outcomes) [2] . Using standard terminology, we will refer to $\mathbf{P}$ interchangeably as a *box* or as *correlations*.

When a Bell game is played, it will in general be found that

$$p(ab|xy) \neq p(a|x)p(b|y)$$

implying that the distant players' outcomes are not statistically independent from each other. Since Alice and Bob may have been in contact in the past, one can expect to find a set of past factors, which can be represented by some shared random variable λ, accounting for such distant statistical dependencies. A *local* explanation (or, model) for the observed probabilities $p(ab|xy)$ consists of the identification of a set of hidden variables which, when taken into account, allow for statistical dependencies between the players outputs (if any) to decouple. In general, the value of such hidden variable will not be constant across different rounds of the game, so we let it vary according to some probability distribution $q(\lambda)$. We will say that a box $\mathbf{P}$ is *local* if it can be written as follows [2]:

$$p(ab|xy) = \int_{\lambda} \mathrm{d}\lambda q(\lambda) p(a|x,\lambda) p(b|y,\lambda) \qquad \forall a,b,x,y. \tag{2.2}$$

Notice that in this picture, we are assuming the *freedom of choice* which states that Alice and Bob are free to choose measurements x and y - thus, λ can not affect and is not affected by system measurements choices.

Quantum correlations – Let's study the case in which Alice and Bob each has a part of a bipartite quantum system which was prepared when they were in contact. This can be two *qudit*, one held by each party, whose joint state we describe, in general, with a mixed state ρ_{AB}, represented in the quantum formalism by a density matrix operator acting on the tensor product Hilbert space $\mathcal{H}^{\mathcal{A}} \otimes \mathcal{H}^{\mathcal{B}}$, where $\mathcal{H}^{\mathcal{A}}$ ($\mathcal{H}^{\mathcal{B}}$) is the Hilbert space where Alice's (Bob's) qudit is lying.
We will say that a box $\mathbf{P}$ is *quantum* if the probabilities are given by Born's rule, i.e. if

$$p(ab|xy) = \mathrm{Tr}(\rho_{AB} M_x^a \otimes M_y^b) \qquad \forall a,b,x,y \tag{2.3}$$

[2] In a classic Bell game (or CHSH scenario), Alice and Bob have two inputs and two outcomes $a,b,x,y \in \{0,1\}$, $\mathbf{P}$ is so expressed as the set of the 16 possible combinations of a,b,x,y conditional probabilities:

$$\mathbf{P} = (p(00|00), p(01|00), \ldots, p(11|11)). \tag{2.1}$$

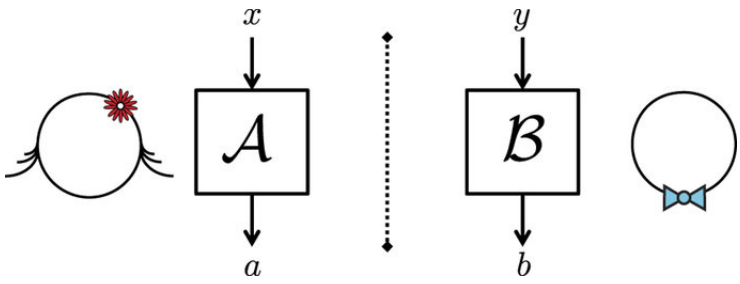

Figure 2.1: A schematic representation of a Bell game with two party, Alice and Bob. Alice (Bob) receive an input x (y) and performs the corresponding measurement on her states. The outcome is labelled a (b).

where $\{M_x^a\}$ (resp. $\{M_y^b\}$), are POVMs (Positive-Operator Valued Measurements)[3] on $\mathcal{H}_A$ (resp. $\mathcal{H}_B$) characterizing Alice's (resp. Bob's) measurements. For some specific quantum states[4] and measurements[5], one can obtain a quantum box $\mathbf{P}$ that doesn't admit a decomposition of the form (2.2) and thus can never be described by a *local hidden variable* model. Hereafter, the term *nonlocality* will be used to refer to such correlations.

Non-signaling correlations – If one wants to explain correlations with a minimal set of assumptions, one can study *non-signaling* correlations [17]. These correlations are generated when the only constraint is : a party's local experiment do not depend on other party's local experiment. In other words, this states that the local marginal probabilities of one party are independent of the other party's measurement setting. More formally, a non-signaling box $\mathbf{P}$ is one for which

$$\begin{aligned} \sum_{b=1}^{d_b} p(ab|xy) &= \sum_{b=1}^{d_b} p(ab|xy') = p(a|x), \quad \forall a, x, y, y' \\ \sum_{a=1}^{d_a} p(ab|xy) &= \sum_{a=1}^{d_a} p(ab|x'y) = p(b|y), \quad \forall b, y, x, x' \end{aligned} \tag{2.4}$$

[3]POVM are define as a decomposition of a k-dimensional Haar unitary matrix set into a set of d positive semidefinite operators, $\{M_x^a\}_{a=1}^d$, where d in the number of outcome, and such that $M_x^a \geq 0 \, \forall a \in \{1, \ldots, d\}$, and $\sum_{a=1}^d M_x^a = \mathbb{1}_k$

[4]With *separable* quantum states, correlations obtained by (Eq. 2.3) are always local. In order to have nonlocality, entangled states are necessary but not sufficient.

[5]If one of the measurement performed locally is *jointly measurable*, correlations generated are all local.

Local correlations trivially satisfy the above constraints. This is also the case for quantum correlations, because

$$\begin{aligned}\sum_b p(ab|xy) &= \mathrm{Tr}(\rho_{AB} M_x^a \otimes (\sum_b M_y^b)) \\ &= \mathrm{Tr}(\rho_{AB} M_x^a \otimes \mathbb{1}) \\ &= \mathrm{Tr}(\rho_A M_x^a) \\ &= p(a|x)\end{aligned} \tag{2.5}$$

and similarly for $p(b|y)$.

2.2 Local Polytope

The local polytope – The set of local distributions – admitting a decomposition of the form (2.2) – is the so-called *local set*, denoted by $\mathcal{L}$. We consider the two-partite (or bipartite) Bell game described above. A deterministic local box is characterized by a strategy in which any given party (Alice or Bob) for any given input always gives the same outcome. Formally:

$$\mathbf{P}_D = \{p_D(ab|xy)\} = \{D(a|x)D(b|y)\} \tag{2.6}$$

with $D(a|x) \in \{0, 1\}$, a deterministic probability distribution (same for $D(b|y)$). We will denote the set of deterministic local boxes with $\mathcal{L}_{det}$. It is not hard to see that any local box $\mathbf{P}$ can always be expressed as convex combinations of deterministic ones

$$\mathbf{P} = \sum_{\mathbf{P}_D \in \mathcal{L}_{det}} c_{\mathbf{P}_D} \mathbf{P}_D, \quad \text{with} \quad c_{\mathbf{P}_D} \geq 0, \sum_{\mathbf{P}_D \in \mathcal{L}_{det}} c_{\mathbf{P}_D} = 1. \tag{2.7}$$

Geometrically speaking, this implies that the set $\mathcal{L}$ forms a *polytope* – a convex set with a finite number of extremal points (or, vertices). One can notice, in the case where Alice and Bob share the same number of measurements m and outputs d, that the number of vertices of the corresponding local polytope $\mathcal{L}$ is d^{2m} [18].
Using the Minkowski-Weyl's theorem, we can also see the local polytope as the intersection of finitely many half-spaces [19],

$$\mathcal{L} = \{\mathbf{P} \mid \mathbf{b}^i \cdot \mathbf{P} \geq \mathbf{b}_0^i, \ \forall i \in \mathcal{I}\} \tag{2.8}$$

with $\mathbf{b}_i \in \mathbb{R}^{d_A d_B m_A m_B}$, $\mathbf{b}^i \cdot \mathbf{P} := \sum_{a,b,x,y} b^i(a,b,x,y) p(ab|xy)$ and $\{\mathbf{b}_i \cdot \mathbf{P} \geq \mathbf{b}_0^i, \ i \in \mathcal{I}\}$ a finite set of inequalities. Those inequalities (or half-spaces)

are the so-called *Bell inequalities* [2]. One can deduce that if a correlation is nonlocal, it has to *violate* some Bell inequality, and thus, be outside the local polytope.

Convex optimisation – The polytope representation of the local set is useful in that provides an algorithm for determining whether a given box is local. This is often refered to as the *polytope membership problem* : given a set of vertices defining a polytope, and given a query point lying in the same space, find whether this point belongs to the polytope. A reformulation of this membership problem, applied to our case, is to determine whether, for a given box $\mathbf{P}$, there exist coefficients c_i satisfying (Eq. 2.7). This reformulation is a typical instance of a *linear programming* (LP) problem [20] – the problem of minimizing or maximizing a linear function over a polyhedron. Specifically, for the problem of deciding the nonlocality of a box $\mathbf{Q}$ we have the optimization problem:

$$\begin{aligned} \min_{\{c_{\mathbf{P}}\}} \quad & 1 \\ \text{s.t.} \quad & \sum_{\mathbf{P}_D \in \mathcal{L}_{det}} c_{\mathbf{P}_D} \mathbf{P}_D = \mathbf{Q}, \\ & \sum_{\mathbf{P}_D \in \mathcal{L}_{det}} c_{\mathbf{P}_D} = 1 \\ & c_{\mathbf{P}_D} \geq 0, \quad \forall \mathbf{P}_D \in \mathcal{L}_{det}. \end{aligned} \tag{2.9}$$

2.3 Locality in Multipartite System

Correlations – Let's describe a multipartite Bell game. In this setting, we have N parties, each having a system with inputs $x_i \in \mathbf{X}$ and outputs $a_i \in \mathbf{A}$ $\forall i \in \{1, \ldots, N\}$. Generalizing the definition of locality (Eq. 2.2) to N parties is straightforward. The local probability distribution representing a local hidden variable model in a multipartite system goes as follow:

$$p(\mathbf{a}|\mathbf{x}) = \int_{\lambda} \mathrm{d}\lambda q(\lambda) \prod_{i=1}^{N} p(a_i|x_i, \lambda) \tag{2.10}$$

where $\mathbf{a} \in \mathbf{A}^N$ (resp. $\mathbf{x} \in \mathbf{X}^N$) are vectors containing the outputs (resp. inputs) of the N parties. Thereby, a local box in a multipartite system is a box that admits a decomposition of the form in (Eq. 2.10).

In the same way, one can also generalize the notion of non-signaling to a multipartite system. We need to add here that the non-signaling con-

straint should be respected by all bipartitions of the N parties. Denoting by $(b, \bar{b}) \in \mathcal{B}$ an arbitrary bipartition of the N-parties, multipartite non-signaling constraints state that

$$\sum_{\mathbf{a}_{\bar{b}}} p(\mathbf{a}_b, \mathbf{a}_{\bar{b}}|\mathbf{x}_b, \mathbf{x}_{\bar{b}}) = \sum_{\mathbf{a}_{\bar{b}}} p(\mathbf{a}_b, \mathbf{a}_{\bar{b}}|\mathbf{x}_b, \mathbf{x'}_{\bar{b}}) = p(\mathbf{a}_b|\mathbf{x}_b), \tag{2.11}$$

with $\mathbf{a}_b, \mathbf{x}_b$ the inputs and outputs of the bipartition b (equivalently for $\bar{b}$).

Generalizing the notion of nonlocality to multipartite system is not trivial. One can use the same method we used previously and state that multipartite nonlocality correspond to boxes that do not admit a decomposition of the type (Eq. 2.10). However, this is a weak definition of nonlocality because, as noticed by Svetlichny [21] in the tripartite case and generalized to N-partites in [22], one can find a bi-separation of (Eq. 2.10); conserving the notation of (Eq. 2.11), such a separation can be written in the form bellow [23],

$$p(\mathbf{a}|\mathbf{x}) = \sum_{(b,\bar{b})\in\mathcal{B}} p_b \int_\lambda \mathrm{d}\lambda q^b(\lambda) p(\mathbf{a}_b|\mathbf{x}_b) p(\mathbf{a}_{\bar{b}}|\mathbf{x}_{\bar{b}}). \tag{2.12}$$

Thus, a stronger definition of multipartite nonlocality considers boxes which do not admit a decomposition of the form in (Eq. 2.12), or that are not *bi-separable*. This is referred to as *genuine multipartite nonlocality* (GMNL) [24].

Multipartite local polytope – As in the bipartite case (see 2.2), one can define the multipartite local set using multipartite deterministic local strategies (Eq. 2.13)

$$\mathbf{P}_D = \{p_D(\mathbf{a}|\mathbf{x})\} = \{D(a_1|x_1)D(a_2|x_2)\dots D(a_n|x_n)\}. \tag{2.13}$$

Indeed, a multipartite local box, in the sense presented above in (Eq. 2.10), can be described as a convex combination of deterministic boxes. The local set can thus be written as,

$$\mathcal{L} = \{\mathbf{P} \mid \mathbf{P} = \sum_i c_i \mathbf{P}_D^{(i)},\ \forall c_i \geq 0, \text{ s.t.} \sum_i c_i = 1\}. \tag{2.14}$$

(Eq. 2.14) still describes a polytope. The mindful reader may notice that the number of vertices characterizing this multipartite local polytope grows exponentially with the number of parties (d^{Nm}). Therefore, even

the most efficient algorithms known to date for the polytope membership problem, whose time complexity is polynomial in the number of vertices (see e.g. [25]), will have a running time exponential in N on this polytopes. In addition, notice that, already for the simple scenario of two parties with m dichotomic measurements, deciding locality is an NP-complete problem [26].

2.4 Symmetrized Local Polytope

2.4.1 Symmetrized Space

Recently, it has been shown that bi-separable nonlocality can be detected in many body-systems using correlators only up to the second order [15]. More specifically, the method introduced in [15] provides Bell inequalities involving only single-body and two-body correlators to detect nonlocality in the multipartite setting.
The setup is the same as explained in Section 2.3 : N parties, with inputs $\mathbf{x} := (x(1), \cdots, x(N)) \in \mathbf{X}^N$, and outputs $\mathbf{a} := (a(1), \cdots, a(N)) \in \mathbf{A}^N$. We define the symmetrized single-body correlators (or local expectation values),

$$S_x := \sum_{i=1}^{N} \langle A_x^{(i)} \rangle, \quad \forall x \in \{1, \ldots, m\} \tag{2.15}$$

with

$$\langle A_x^{(i)} \rangle := \sum_{\mathbf{a} \in \mathbf{A}^N} \mathbf{a}(i) \cdot p(\mathbf{a} | x_1, \ldots, x, \ldots, x_N),$$

and where we assume nonsignaling constraints (so the i^{th} party's marginal distribution for input x is independent of the choice of inputs for the other parties). Similarly, we have the symmetrized two-body correlators:

$$S_{xy} := \sum_{\substack{i,j=1 \\ i \neq j}}^{N} \langle A_x^{(i)} A_y^{(j)} \rangle, \quad \forall x, y \in \{1, \ldots, m\}. \tag{2.16}$$

with

$$\langle A_x^{(i)} A_y^{(j)} \rangle := \sum_{\mathbf{a} \in \mathbf{A}^N} \mathbf{a}(i)\mathbf{a}(j) \cdot p(\mathbf{a} | x_1, \ldots, x_i, \ldots, x_j, \ldots, x_N).$$

We can now consider a symmetrized space, determined by five-dimensional vectors

$$(S_0, S_1, S_{00}, S_{01}, S_{11}). \tag{2.17}$$

As one can notice, the number of parties is absorbed in the definition of body correlators ; *we are now dealing with a space whose dimension is independent of* N.

2.4.2 Symmetrized Polytope

Deterministic correlators – One can compute single-body and two-body correlators for every deterministic strategy. Here we present the method of [15], for a two-input, $\{0,1\}$, two-output, $\{-1,1\}$, and N-partite setting. First we define the number of parties playing each of the four deterministic strategy,

$$\begin{aligned} a &:= \#\{i \in \{1,\dots,N\} \mid a_0^{(i)} = 1, a_1^{(i)} = 1\} \\ b &:= \#\{i \in \{1,\dots,N\} \mid a_0^{(i)} = 1, a_1^{(i)} = -1\} \\ c &:= \#\{i \in \{1,\dots,N\} \mid a_0^{(i)} = -1, a_1^{(i)} = 1\} \\ d &:= \#\{i \in \{1,\dots,N\} \mid a_0^{(i)} = -1, a_1^{(i)} = -1\}. \end{aligned} \tag{2.18}$$

With the trivial condition $a+b+c+d = N$. One can then rewrite correlators,

$$\begin{aligned} S_0 &= a + b - c - d \\ S_1 &= a - b + c - d \\ S_{00} &= S_0^2 - N \\ S_{01} &= S_0 S_1 - (a - b - c + d) \\ S_{11} &= S_1^2 - N \end{aligned} \tag{2.19}$$

Thus, single-body and two-body correlators for deterministic strategies can be parametrized by elements in the set $\{(a,b,c,d) \in \mathbb{N}^4 \mid a+b+c+d = N\}$ which is isomorphic to a 3-tetrahedron,

$$\mathbb{T}_N = \{(a,b,c) \in \mathbb{N}^3 \mid a+b+c \leq N\}. \tag{2.20}$$

Tura et al. [15] proved that the deterministic symmetrized polytope we want to describe is only represented by 4-tuples that belong to the boundary of the above tetrahedron, $\partial\mathbb{T}_N$. This is,

$$\partial\mathbb{T}_N = \{(a,b,c,d) \in \mathbb{N}^4 \mid a+b+c+d = N,\, abcd = 0\} \tag{2.21}$$

We generalized this method to an arbitrary number of inputs m. The

quantities of (Eq. 2.18) can be written as,

$$\begin{aligned}
\alpha_1 &= \#\{i \in \{1,\dots,N\} \mid a_1^{(i)} = 1, a_2^{(i)} = 1, \dots, a_{m-1}^{(i)} = 1, a_m^{(i)} = 1\} \\
\alpha_2 &= \#\{i \in \{1,\dots,N\} \mid a_1^{(i)} = 1, a_2^{(i)} = 1, \dots, a_{m-1}^{(i)} = 1, a_m^{(i)} = -1\} \\
\alpha_3 &= \#\{i \in \{1,\dots,N\} \mid a_1^{(i)} = 1, a_2^{(i)} = 1, \dots, a_{m-1}^{(i)} = -1, a_m^{(i)} = 1\} \\
\alpha_4 &= \#\{i \in \{1,\dots,N\} \mid a_1^{(i)} = 1, a_2^{(i)} = 1, \dots, a_{m-1}^{(i)} = -1, a_m^{(i)} = -1\} \\
&\vdots \\
\alpha_{2^m} &= \#\{i \in \{1,\dots,N\} \mid a_1^{(i)} = -1, a_2^{(i)} = -1, \dots, a_{m-1}^{(i)} = -1, a_m^{(i)} = -1\}
\end{aligned} \tag{2.22}$$

Where α_n correspond to the number of parties with a deterministic strategy which output a string $(a_1, \dots, a_m)$ equivalent to the binary expression of the number $n-1$, in which 1 replace 0 and -1 replace 1. Also, the condition $\sum_{i=1}^{2^m} \alpha_i = N$ holds. We can rewrite the above expression in a matrix form:

$$A = \begin{bmatrix}
\alpha_1 & 1 & 1 & \dots & 1 & 1 \\
\alpha_2 & 1 & 1 & \dots & 1 & -1 \\
\alpha_3 & 1 & 1 & \dots & -1 & 1 \\
\alpha_4 & 1 & 1 & \dots & -1 & -1 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\alpha_{2^m-1} & -1 & -1 & \dots & -1 & 1 \\
\alpha_{2^m} & -1 & -1 & \dots & -1 & -1
\end{bmatrix} \tag{2.23}$$

Hence, single body correlator for an input $k \in \{1, \dots, m\}$ is of the form,

$$S_k = \sum_{i=1}^{2^m} A_{i,0} A_{i,k} \tag{2.24}$$

where $A_{i,j}$ is the element on the i^{th} line, and on the j^{th} column, of matrix A define in (Eq. 2.23). And two body correlators are,

$$S_{kl} = S_k S_l - \sum_{i=1}^{2^m} A_{i,0} A_{i,k} A_{i,l} \tag{2.25}$$

As with Tura et al.'s method, in order to find every deterministic body correlators, we just need to find all 2^m-tuple in the set,

$$\mathbb{S} = \{\boldsymbol{\alpha} \in N^{2^m} \mid \sum_{i=1}^{2^m} \alpha_i = N, \exists \alpha_i = 0 \quad \forall i \in \{1, \dots, 2^m\}\}. \tag{2.26}$$

Characteristics of the symmetrized polytope – The space $(S_0, S_1, S_{00}, S_{01}, S_{11})$ of one and two-body correlators in a given multipartite game is a subspace of boxes $P(\mathbf{a}|\mathbf{x})$ in this same game. Therefore, one can define the symmetrized local polytope as the convex hull of every one and two-body correlator constructed from a deterministic strategy (thus, from deterministic boxes),

$$\mathbb{P}^S = \mathrm{Conv}(\{(S_0, S_1, S_{00}, S_{01}, S_{11})_D\}). \tag{2.27}$$

The cardinality of the set defined in (Eq. 2.26) can be easily seen to be $2(N^m + 1)$. Thus, the polytope $\mathbb{P}^S$ have a number of vertices which grows polynomially with degree m. Linear programming can thus be used on such polytopes to determine the locality of a given correlation[6] for reasonably small values of N and m.

[6]Projecting a correlation to the symmetrized space leads to information loss. Hence, some nonlocal boxes, once projected into the symmetrized space, might be indistinguishable from local ones. Only nonlocal correlations originating from permutationally invariant quantum states and measurements are guarantee to be found nonlocal in $\mathbb{P}^S$.

Chapter 3

Machine Learning for Nonlocality Detection in Multipartite System

Machine learning, a subfield of artificial intelligence, represents algorithms based on statistical models to give computers the ability to learn from and make predictions on data [27]. In this chapter, after a brief introduction to the basic concepts of the theory, we report on how we used this learning paradigm to test nonlocality in the scenario described in the preceding chapter.

3.1 Neural Networks

(Artificial) Neural networks (NN) are a subclass of machine learning algorithms. They consist of a network of interconnected *artificial neurons* that take some inputs and process them to perform a task, e.g. making prediction on new data, generating similar data, compressing data, etc.

In this section, we will focus on NN for binary classification, since our aim is to determine whether a correlation is local or nonlocal. As an introduction, *Perceptron* and *Adaline*, two single-neurone algorithms, will be presented. We will then introduce *Feedforward Neural Networks*, a widely used type of NN for nonlinear classification. Finally, Domain-Adversarial Neural Network, a NN for classification of data originating from different domains, will be explained.

3.1.1 Introduction : Perceptron and Adaline

Perceptron – A perceptron (Fig. 3.1) is a biomimic mathematical model for linear classification, inspired by signal processing between neural cells that are assumed to be either *active* or *resting* [28]. It is an instance of supervised learning, a class of machine learning where an algorithm learns how to process inputs to outputs with examples of input-output pairs. The perceptron model is used for binary classification – mapping inputs to either 0 or 1 – in the case of linearly separable classes, or *labels* [29].

Let's describe the perceptron algorithm [30]. First, the perceptron takes a vector of inputs $\mathbf{x}$ where each element is called a *feature*. To every feature a weight is assigned – weights can also be represented as a vector $\mathbf{w}$. These weights are first initialized randomly. We can now compute the *net input* z,

$$z := \mathbf{w}^T \mathbf{x}. \tag{3.1}$$

Then a *decision function* $\Phi(\cdot)$, or threshold, is applied to the net input. It is similar to a unit step function,

$$\Phi(z) := \begin{cases} 1, & \text{if } z \geq \theta \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

where θ is a fixed bias that shifts the decision boundary from the origin. The bias can be absorbed in $\mathbf{w}$ for convenience (with a corresponding input x_θ being always 1). This function maps the net input to a specific label : this is the perceptron's prediction, y'.
Weights can now be updated *via* the *backpropagation* of an error. For a perceptron, this error is defined as the difference between the prediction and the desired label, y. Thus, with this error, on the i^{th} learning sample the weight will be updated as

$$w_j = w_j + \Delta w_j \tag{3.3}$$

with

$$\Delta w_j = (y^{(i)} - y'^{(i)}) x_j^{(i)}, \tag{3.4}$$

where j is the j^{th} feature, x_j its value and w_j its weight.
Training a perceptron consists of iterating the above algorithm on a learning set to *in fine* converge to the best reachable classification. Indeed, with such an algorithm, only linearly separable data can be perfectly classified [31].

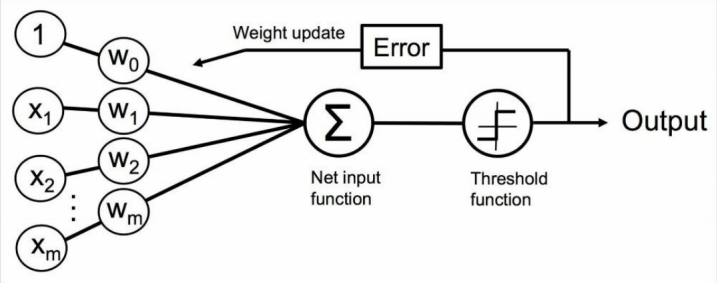

Figure 3.1: A schematic representation of a perceptron.

Adaline – An important improvement to the perceptron was the *Adaline* [32] (**Ada**ptative **Li**near **Ne**urone) algorithm. It is based on the perceptron algorithm with the addition of an *activation function* between the net input computation step and the threshold one (Fig. 3.2). This function can be nonlinear. The key point is that the error is now computed from the activation function instead of an unit step like function. We denote this activation function with $\phi(\cdot)$.
Consider an Adaline with m inputs. With the use of a cost function, $C(\cdot)$, from which the error will be computed, we have the following updating rules:

$$\mathbf{w} = \mathbf{w} + \Delta\mathbf{w} \tag{3.5}$$

where $\Delta\mathbf{w}$ is the vector of weights change $\Delta\mathbf{w} = (\Delta w_1, \ldots, \Delta w_m)$. Since our objective is to optimize the classification accuracy, thus minimizing the global cost, weight's changes can be defined as the negative gradient of the cost function (multiply by a learning rate, η),

$$\Delta\mathbf{w} = -\eta\nabla C(\phi(\mathbf{w})), \tag{3.6}$$

where ∇C is the gradient of C on every weights, or,

$$\nabla C = (\frac{\partial C}{\partial w_1}, \ldots, \frac{\partial C}{\partial w_m}). \tag{3.7}$$

The use of an activation function allows for better classification of non trivially separable data and gives the opportunity of using a learning rate[1] and of setting a custom cost function.

[1] Using a learning rate in the perceptron algorithm is similar to a rescaling of weights, thus it does not qualify as a proper learning rate.

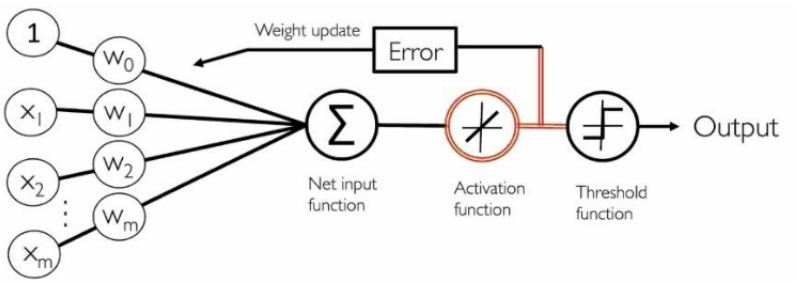

Figure 3.2: A schematic representation of an Adaline.

3.1.2 Feedforward Neural Network - FNN

Feedforward neural network is a type of neural network, used for classification of nonlinearly separable data. It consists in a succession of, at least, three layers of neurones. Each layer contains a fix number of neurones, and each neurone receives inputs from all neurones of the preceding layer (Fig. 3.3), explaining this denomination.

The first layer of a neural network is the *input layer*. The number of neurones in this first layer is fixed by data dimension. It follows one or more *hidden layers*. The more they are, the more accurate classification of highly non linear data will be, but the heavier in computational resources training the network will get. Finally, the *output layer* contains as many neurones as classes.
Each neurone is similar to an Adaline : it takes inputs and computes the net input, on which an activation function $\phi(\cdot)$ is then applied. A cost function $C(\cdot)$ is also defined.

We consider a FNN of L layers. A single weight is denote by w^l_{jk} : the weight from the k^{th} neurone in the $(l-1)^{\text{th}}$ layer to the j^{th} neurone in the l^{th} layer.
Training a FNN consists of minimizing the cost, therefore, updating weights as in (Eq. 3.6). Thus, we need to compute the gradient of the cost for every weight $\frac{\partial C}{\partial w^l_{jk}}$. This is done using the *backpropagation* algorithm [33]. A formulation of this algorithm is as follows [34]:

1. **Input**: $\mathbf{x}$ set the activation of the first layer, $\mathbf{a}^1$.
2. **Feedforward**: Compute net inputs and activations for each $l \in \{2, \ldots, L\}$:

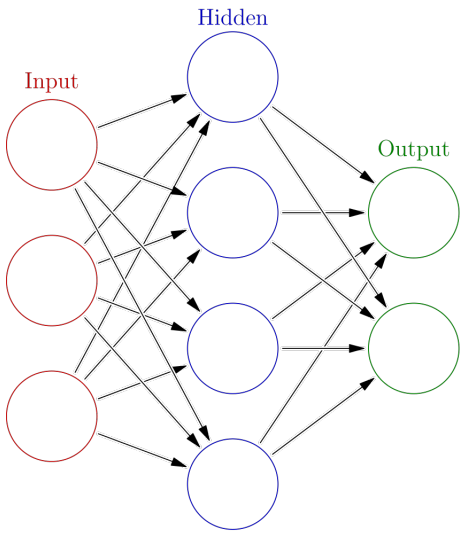

Figure 3.3: A schematic representation of a feedforward neural network with one hidden layer.

$\mathbf{z}^l = (\mathbf{w}^l)^T \mathbf{a}^{l-1}$ and $\mathbf{a}^l = \phi(\mathbf{z}^l)$

3. **Output error**: Compute

$$\boldsymbol{\delta}^L = \nabla_{\mathbf{a}^L} C \odot \phi'(\mathbf{z}^L). \tag{3.8}$$

where, $\odot$ is the Hadamard product and ϕ' is the derivative of ϕ.

4. **Backpropagate the error**: For $l \in \{L-1, L-2, \ldots, 2\}$, compute the error using the recurrence rule,

$$\boldsymbol{\delta}^l = ((\mathbf{w}^{l+1})^T \boldsymbol{\delta}^{l+1} \odot \phi'(\mathbf{z}^l). \tag{3.9}$$

5. **Return**: the gradient of the cost, with each element define as,

$$\frac{\partial C}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j. \tag{3.10}$$

This algorithm can be speed up by a *stochastic gradient descent* (SGD) algorithm (for more details see [35]).

3.1.3 Domain Adversarial Neural Network - DANN

Domain adaptation is the subfield of machine learning that focus on algorithms for learning, from a *source* data distribution, a model that can be extend to a *target* data distribution [36]. Domain adaptation of neural network is a fast-expending field.

Domain Adversarial Neural Network [16] (DANN) is a neural network for domain adaptation using adversarial learning. It consists of three FNN, referred to as *feature extractor* (FE), *domain classifier* (DC), and *label predictor* (yP) (Fig. 3.4). The aim is to extract a new feature space *via* the feature extractor allowing a good label classification while being domain-invariant.

DANN algorithm can be seen as:

1. **Input** : One sample of a dataset containing n samples. Each sample is $S_i = (\mathbf{x}_i, y_i, d_i)$, $\forall i \in \{1, \ldots, n\}$, where $\mathbf{x}_i$ is an input vector, y_i the desired label, and d_i the domain this sample originate from.

2. **Feedforward 1**: Compute activation of every neurones of the FE, up to the last layer. Outputs of the last layer are new features, elements of the *extracted* feature space. Those new features are inputs of the DC and yP.

3. **Feedforward 2**: Compute activation of DC and yP neurones.

4. **Output error**: Compute error of the DC, δ_{DC}, and of the yP, δ_{yP}, using (Eq. 3.8).

5. **Backpropagation 1**: Backpropagate δ_{DC} (δ_{yP}) using (Eq. **??**) through DC (yP).

6. **Backpropagation 2**: Backpropagate δ_{yP} through FE as if it is a single FNN. Thus, the extracted features are updated such that yP performs better at each step.
   Backpropagate $-\lambda\delta_{DC}$ through FE, where $-\lambda$ is a cost weight. Error reversal ensures that the extracted featured distribution over domains are made similar. In other words, extracted features are updated such that it is harder for DC to classify domains.

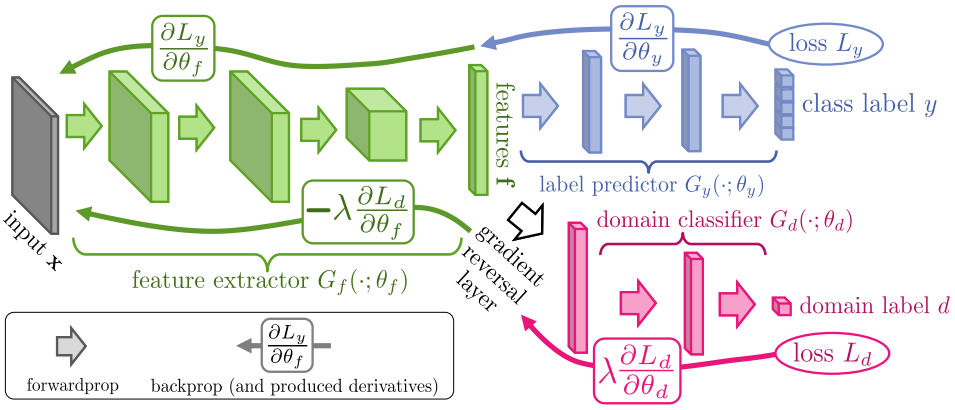

Figure 3.4: A schematic representation of a DANN.

3.2 Generating Data : Local and Nonlocal Correlations

For our supervised learning approach, examples of input-output pairs (in our setting, boxes labelled as local or nonlocal) are mandatory. In this section we explain the strategies we followed to generate them.

3.2.1 Local Correlations: Uniform Sampling of a Convex Polytope

By virtue of its geometric structure, the problem of generating uniform samples from the set of local correlations reduces to the problem of generating uniform samples from a convex polytope. This last problem has been vastly studied in the past 25 years (see, e.g., [37] for a review) and the state-of-the-art strategies to solve it are the Hit-and-Run algorithm of Kannan [38] and the more standard method of Rejection Sampling. In this section we describe these techniques and explain how we applied them to our setting.

Hit-and-Run algorithm – This sampling algorithm is an instance of the class of Markov Chain Monte Carlo (MCMC) [38] algorithms. MCMC methods usually start with a random point in the target space and then "walk" to a next point according to some rule. The Markovianity comes from the fact that each next step depends only on the current location and not on the steps taken previously; it is Monte Carlo because this choice is made pseudorandomly.

The HIT-AND-RUN algorithm (Alg. 1) was introduced by Vempala in [37][2]. Asymptotically in the number of iterations, it provably generates uniform samples of a given convex polytope. However, the convergence rate of the HIT-AND-RUN algorithm gets slower with increasing dimension of the polytope. Convergence of MCMC methods is usually evaluated by how much overlap (or, *mixing*) there is between different (independent) walks (or, chains). To achieve good mixing, we must let the individual chains run for sufficiently long times (i.e. sample more points).

Algorithm 1: HIT-AND-RUN algorithm

Result: A list of N points uniformly distributed in a polytope.
n, the number of generated point, set to 0;
x_n a starting point in a polytope $\mathbb{P} \in \mathbb{R}^k$;
while $n < N$ **do**
  Take a random direction d in the sphere S^k;
  Find the chord C through x_n in direction d and $-d$;
  Using LP, find c_1,c_2, the two intersections of C and $\mathbb{P}$ edges;
  Parametrize the chord as $C = c_1 + \lambda(c_2 - c_1),\ \forall\lambda \in [0; 1]$;
  Pick a random point x_{n+1} on C by taking $\lambda \in U[0; 1]$;
  Increment n by 1;
end

Hit-and-Run sampling of $\mathbb{P}^S$ – A sampling of a symmetrized polytope $\mathbb{P}^S$ with the HIT-AND-RUN algorithm can be seen in (Fig. 3.5). We observed clusters of points in some extreme regions of the polytope (red circle in Fig. 3.5). This might be due to the "sharpness" of some of the symmetrized polytope angles, where the chain gets stuck. Furthermore, even with ten thousand points we do not reach a uniform distribution (e.g. the quantity S_{00} in Fig. 3.5).

Rejection sampling – Another method for convex polytope sampling is to use a *rejection sampling* algorithm (Alg. 2), a Monte Carlo algorithm[3]. It consists of, given a polytope $\mathbb{P}$, uniformly generating a point in the smallest hypercube enclosing $\mathbb{P}$ and keeping it or *rejecting* it based on its membership to $\mathbb{P}$.

[2]We provide an implementation of this algorithm in python, see `https://plut0n.gitlab.io/MLNL/_modules/nonloc_tb.html#hit_n_run`.

[3]An implementation in python of this algorithm can be found at `https://plut0n.gitlab.io/MLNL/_modules/nonloc_tb.html#rejection`.

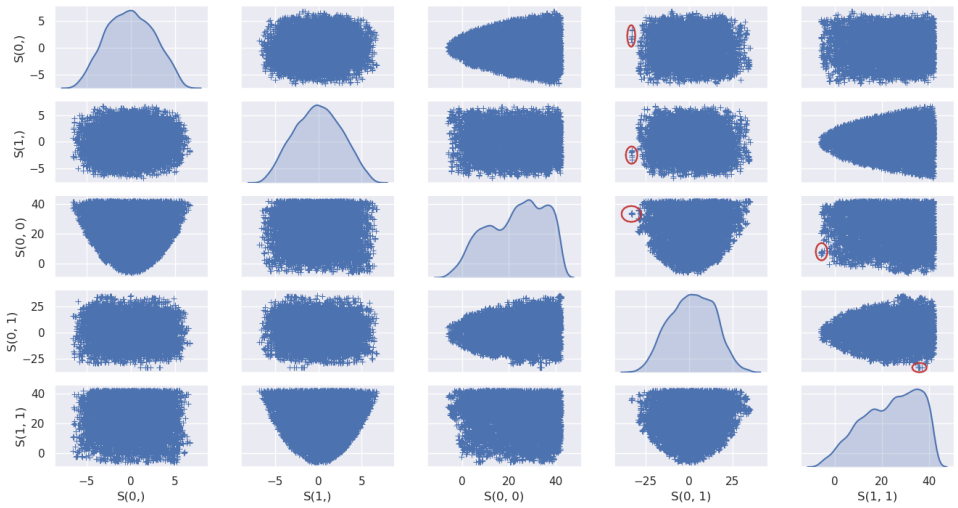

Figure 3.5: Ten thousand points sample, *via* HIT-AND-RUN algorithm, from a symmetrized polytope drawn for 7 parties, 2 inputs and 2 outputs. Here are two dimensional projections in each pair of vectors of (Eq. 2.17). On the diagonal is the kernel density estimate (KDE), an approximation of the probability density function of a random variable. Red circles correspond to clusters of points, where the chain gets stuck for a certain number of steps.

Algorithm 2: Rejection sampling algorithm

Result: A list of N points uniformly distributed in a polytope.
n, the number of generated points, set to 0;
$\mathbb{P}$, the polytope we want to sample;
while $n < N$ **do**
  for $j < dim(\mathbb{P})$ **do**
    Let $a := \min\{p(j)|p \in \mathbb{P}\}$ and $b := \max\{p(j)|p \in \mathbb{P}\}$;
    Generate uniformly x_n, with j^{th} element $x_n^{(j)} = U[a; b]$;
  end
  Using LP check whether x_n is in $\mathbb{P}$;
  if $x_n \in \mathbb{P}$ **then**
    Increment n by 1;
  end
end

Rejection sampling of $\mathbb{P}^S$ – To compute an hypercube enclosing $\mathbb{P}^S$, we need extreme values of its vertices in each direction. Using (Eq. 2.24) and (Eq. 2.25), for N parties, we have

$$S_k \in [-N; N],\ S_{kk} \in [N; N(N-1)],\ S_{kl} \in [-N(N-1); N(N-1)]. \quad (3.11)$$

A sample of $\mathbb{P}^S$ using rejection sampling in the space defined above can be seen in (Fig. 3.6).

Compared to the Hit-and-Run algorithm, rejection sampling exhibits a larger running time over the same polytope $\mathbb{P}^S$. This is explained by the number of LP instances that need to be solved. Hit-and-Run needs a fix number of two LP per generated point, whereas for the rejection sampling this depends on the measure of the polytope inside the enclosing hypercube. On the other hand, the distribution we get with rejection sampling is closer to the uniform one. Since we wanted our machine learning algorithm to be as accurate as possible, we decided to favour this last sampling method for generating local correlations.

3.2.2 Nonlocal Correlations

For the problem of generating nonlocal correlations, we considered two strategies: 1) generating random instances of a class of quantum correlations which we know not to belong to $\mathbb{P}$ by the work of Tura et al. [15] and 2) doing rejection sampling for the complement (inside the space of probabilities) of $\mathbb{P}$.

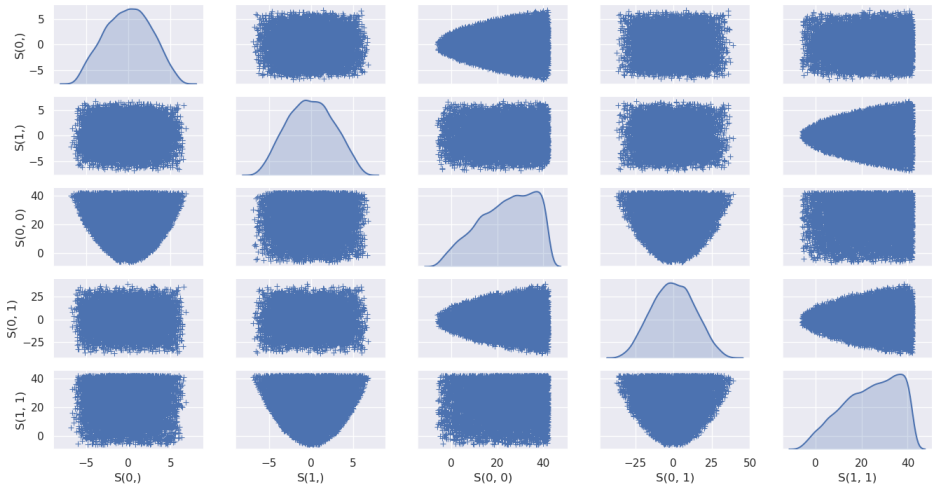

Figure 3.6: Ten thousands points sampled, *via* rejection sampling algorithm, from a symmetrized polytope drawn for 7 parties, 2 inputs and 2 outputs. Here are two dimensional projections in each pair of vectors of (Eq. 2.17). On the diagonal is the kernel density estimate (KDE).

Quantum correlations – For a two-inputs, two-outputs scenario, generating nonlocal correlations in the symmetrized space can be done using N-qubit Dicke states [39] with $k = \lceil \frac{N}{2} \rceil$ excitations,

$$|D_k^N\rangle = \frac{1}{\sqrt{\binom{N}{k}}} \sum_j \mathrm{Perm}_j\{|1\rangle^{\otimes k} \otimes |0\rangle^{\otimes (N-k)}\}, \tag{3.12}$$

where $\sum_j \mathrm{Perm}_j\{\cdot\}$ is the sum over all possible permutations, and with measurements,

$$M_0 = \sigma_z, \quad M_1 = \cos(\theta)\sigma_z + \sin(\theta)\sigma_x. \tag{3.13}$$

This setup violates a Bell inequality composed only by single and two-order body correlators [15], for some specific angle θ.
Using this setup, we were able to generate non local correlations as one can see in (Fig. 3.2.2).

Nonlocal correlations from rejection sampling – Another method to obtain nonlocal correlations is to use rejection sampling as described

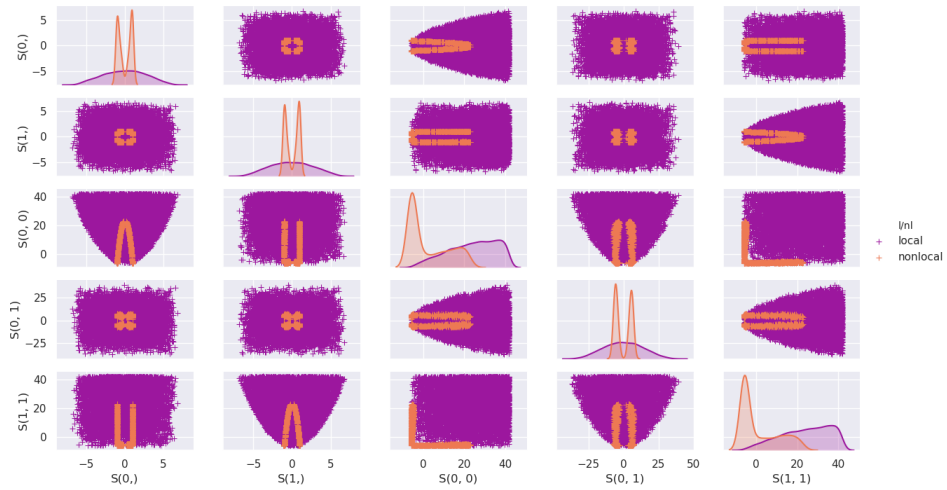

Figure 3.7: Local (purple) and nonlocal (orange) correlations for a 7 parties, 2 inputs and 2 outputs scenario. Local correlations consists of ten thousand points originating from a rejection sampling method. Nonlocal correlations are generating from Dicke states $|D^N_{\lceil \frac{N}{2} \rceil}\rangle$ and using measurements describe above with a random angle $\theta \in U[0; \pi]$. Here are one thousand nonlocal correlations.

in the previous sections. Instead of conserving a point that is found to belong to the polytope, a point is conserved when the LP indicates that it is outside this polytope[4]. Since we want points fully wrapping $\mathbb{P}^S$, thus avoiding common points between the sampling space and $\mathbb{P}^S$ boundaries, we shift boundaries of one and two body correlators defined in (Eq. 3.11) by $-\varepsilon$ for upper bounds and $+\varepsilon$ for lower bounds. Notice that in this way we might get distributions which are not even non-signalling, let alone quantum. The main advantage of such a sampling is to obtain a well define boundary of $\mathbb{P}^S$.

3.3 Results

Our aim is to provide a computationally efficient way to certify the presence of nonlocality in multipartite systems. As explained before, efficiency will come at the cost of having non-perfect accuracy. Since our aim is to certify the presence of nonlocality, we will strive to have as little false positives as possible, that is, local correlations misclassified as nonlocal, while trying to achieve a fairly good accuracy on the classification of nonlocal points. Next, we describe the experimental setup and the results obtained with our methods.

3.3.1 Final Setup

Datasets of correlations – For both local and nonlocal correlations, we generated ten thousand correlations using rejection sampling of the symmetrized polytope $\mathbb{P}^S$, for a 2-inputs, 2-outputs scenario, and for different number of parties $N \in \{3, 4, 5, \ldots, 20, 25, 30, 40, 50, 75, 100\}$[5]. Every correlation is labelled with $y \in \{0, 1\}$, where 0 means local, 1 nonlocal and with d, the number of parties a correlation originate from. Thus, we have a dataset of the form $D = \{(\mathbf{x}_i, y_i, d_i)\}$.

Since the value of the correlators S_x and S_{xy} increases with N and our aim was to have a classification method which is (as much as possible) independent of N, we decided to test what happened if we trained (and tested) our classifiers with unnormalized data; that is, we did experiments

[4] In (Alg. 2), we replace the argument of the if condition, $x \in \mathbb{P}$, by $x \notin \mathbb{P}$.

[5] Our data and a custom dataset loader for pytorch we implemented are available at `https://gitlab.com/plut0n/MLNL/tree/master/data` and `https://plut0n.gitlab.io/MLNL/_modules/data.html#Data_N_File` respectively.

with another dataset $D' = \{(\mathbf{x}'_i, y_i, d_i)\}$ with

$$x'^{(j)}_i = 2\frac{x^{(j)}_i - \min((\mathbb{P}^S)^{(j)})}{\max((\mathbb{P}^S)^{(j)}) - \min((\mathbb{P}^S)^{(j)})} - 1 \tag{3.14}$$

and $\min((\mathbb{P}^S)^{(j)})$ (resp. $\max((\mathbb{P}^S)^{(j)})$) being the minimum (resp. maximum) value a local correlation can take in the basis j, defined in (Eq. 3.11).

FNN – We designed a FNN with three layers of a fixed number of neurones: 5 neurons in the first layer, 10 in the second and 2 in the third. Activation function of the hidden layer is the rectified linear unit, or *ReLU*

$$\text{ReLU}(z) = max(0, z). \tag{3.15}$$

Using ReLU for this layer is motivated by the stability of the gradient when computed from this activation function [6].
Activation function of the output layer is a normalized exponential function, or *Softmax* [40]

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{K} e^{z_k}}, \tag{3.16}$$

where z_i is the net input of the j^{th} neurone, and K is the number of class, thus, the number of neurones in the output layer.
The loss function we used is the *negative log likelihood loss*, or *NLLLoss*. Therefore, the cost is defined by,

$$C = -\sum_i t_i \log(a_i) \tag{3.17}$$

where t_i is the cost weight for the class i, and a_i the activation of the i^{th} neurone (of the last layer). Similarly, one can set the activation function to be linear, and unify *Softmax* and *NLLLoss* in a single cost function. This new cost function is the so-called *cross-entropy* [41]. Conceptually, this loss can be seen as the difference between the distribution of neurones activation, and the "one-shot" distribution of the desired label, a vector of all 0s except a 1 at the neurone corresponding to the desired class.
We use cost weights of 10 for the local class and 1 for the nonlocal class. Those imbalanced weights are equivalent to using imbalanced data : 10 times more local correlations than nonlocal correlations in the dataset. Hence,

[6] Gradient descent might encounter vanishing or exploding problem when computed from other activation such as softmax, or sigmoïd [34].

weights of the neural network will be updated such that the accuracy on local correlations is maximized.

DANN – We implemented a DANN with the following architecture:

- Feature extractor is of three layers with number of neurones per layer being: 5 for the first and second and 3 for the third. Activation function on layer but the input one is ReLU (Eq. 3.15).
- Label predictor has the same architecture than the above FNN, except the input layer that consists of 3 neurones.
- Domain classifier is of three layers with number of neurones per layer being: 3 for the first, 4 for the second and $N_{end} - N_{start}$ for the third, with N_{start} (resp. N_{end}) being the minimum (resp. maximum) number of parties the DANN was trained on, assuming that the DANN is trained on every number of parties between N_{start} and N_{end}. Activation and cost functions are the same as in the above FNN.

3.3.2 FNN Results

Results on unnormalized data – Performance of a FNN trained on correlations originating from 3 to 6-partites systems and run for 200 epochs[7] can be seen in Fig. 3.8. Effect of imbalanced cost weights is clearly visible with a higher local than nonlocal accuracy.
Accuracy of the FNN decreases when we increase the number of parties in the training data. This was expected, as the number of vertices of $\mathbb{P}^S$ grows with N. However, we also observe that the classification accuracy converges for high N. With the assumption that $\mathbb{P}^S$ have really sharp angles, this behaviour might partially be due to a decreasing number of sample points into $\mathbb{P}^S_{N+1} \setminus \mathbb{P}^S_N$, inversely of N. Indeed, with a fixed number of sampled points, using rejection sampling, the higher N is, the sparser points into sharp area of the symmetrized polytope should get. We also need to take into account imbalanced cost weights which may also participate into the explanation of such behaviour: with an higher cost weight for local correlations, FNN will have a tendency to classify correlations, especially when then they lie in a space the networks wasn't trained on, as local.

Results on normalized data – As we expected, normalizing the data helped the FNN to achieve (markedly) better results (see Fig 3.9), because,

[7] An epoch is an iteration over a dataset.

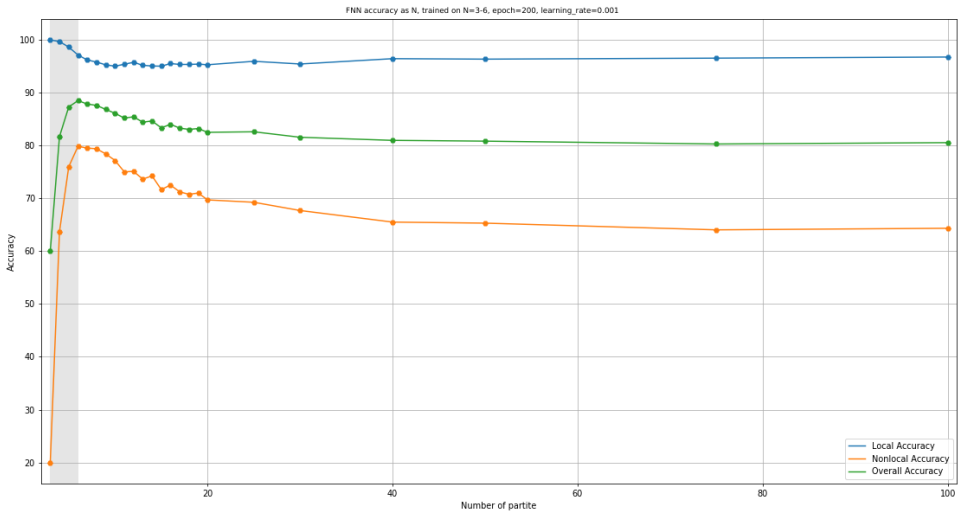

Figure 3.8: FNN accuracy for local (blue) and nonlocal (orange) correlations. Overall accuracy of the network is represented by the green line. Trained for correlations from 3 to 6-partite system. FNN tested on $N \in \{3, 4, 5, \ldots, 20, 25, 30, 40, 50, 75, 100\}$.

in a way, normalizing implies blurring the dependence of the data with the number of parties. Indeed, when plotting 2D-projection of the space of symmetrized correlators in every pair of basis in (Eq. 2.17) and for different N, it appears that already for $N \approx 10$ the shape of the polytopes converge to its final form. Therefore, we decided to test what happened if we run the linear program for testing membership to the symmetrized polytope of $1 \leq N \leq 9$ parties with data from the spaces of $N = 3, \ldots, 100$ parties. As expected, this gives a good classification scheme with an accuracy of more than 0.87 for every tested case (see Fig. 3.10).

3.3.3 DANN Results

We trained the DANN on correlations generated from systems with 3 to 6 parties, with 150 epochs and a learning rate of 10^{-4} [8]. The DANN's accuracy on $N \in \{3, 4, 5, \ldots, 20, 25, 30, 40, 50, 75, 100\}$ can be seen in Fig. 3.11.

[8] We provide a pre-trained model for DANN at `https://gitlab.com/plut0n/MLNL/blob/master/src/dann.pkl`. This model can be load with the *load* function of the class NL_DANN.

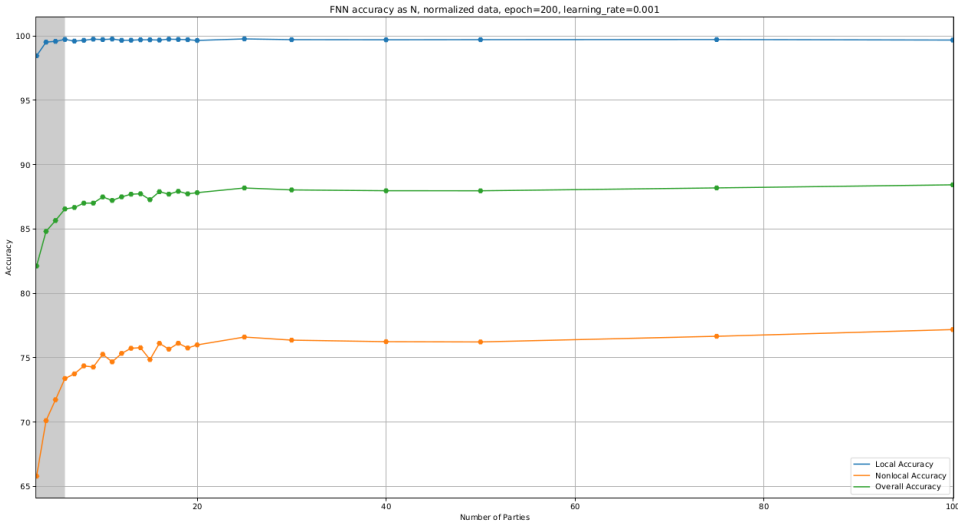

Figure 3.9: Accuracy of FNN trained on normalized correlations from $N \in \{3, 4, 5, 6\}$, with 200 epochs, and a learning rate of 10^{-3}.

N \ P^S_N	3		4		5		6		7		8		9	
	L	NL	L	NL	L	NL	L	NL	L	NL	L	NL	L	NL
3	1.000	1.000	0.946	0.963	0.951	0.963	0.942	0.965	0.936	0.965	0.932	0.967	0.926	0.965
4	0.903	0.975	1.000	1.000	0.965	0.990	0.967	0.987	0.958	0.989	0.956	0.988	0.950	0.988
5	0.905	0.982	0.960	0.988	1.000	1.000	0.981	0.994	0.980	0.997	0.980	0.997	0.972	0.997
6	0.906	0.985	0.965	0.994	0.987	0.995	1.000	1.000	0.989	0.997	0.989	0.997	0.982	0.996
7	0.876	0.982	0.942	0.985	0.974	0.995	0.983	0.994	1.000	1.000	0.985	0.998	0.983	0.998
8	0.894	0.987	0.956	0.992	0.983	0.996	0.993	0.996	0.996	0.997	1.000	1.000	0.998	0.998
9	0.908	0.981	0.950	0.989	0.981	0.991	0.991	0.995	0.991	0.998	0.993	0.997	1.000	1.000
10	0.907	0.984	0.959	0.988	0.983	0.989	0.987	0.993	0.994	0.997	0.994	0.997	0.996	1.000
11	0.890	0.983	0.941	0.988	0.976	0.995	0.977	0.995	0.985	0.997	0.990	0.998	0.993	0.999
12	0.908	0.983	0.966	0.993	0.983	0.993	0.989	0.993	0.996	0.997	0.997	0.997	0.998	0.997
13	0.893	0.984	0.959	0.988	0.971	0.993	0.989	0.991	0.987	0.997	0.993	0.997	0.996	0.999
14	0.895	0.982	0.950	0.985	0.973	0.987	0.985	0.988	0.991	0.989	0.991	0.991	0.994	0.996
15	0.893	0.986	0.952	0.986	0.970	0.989	0.983	0.990	0.994	0.995	0.992	0.997	0.999	0.998
16	0.886	0.980	0.941	0.987	0.972	0.987	0.984	0.991	0.991	0.995	0.991	0.994	0.996	0.996
17	0.899	0.983	0.955	0.986	0.977	0.985	0.983	0.989	0.989	0.992	0.994	0.993	0.993	0.997
18	0.919	0.981	0.954	0.988	0.966	0.990	0.981	0.988	0.986	0.992	0.990	0.994	0.994	0.994
19	0.904	0.977	0.955	0.980	0.970	0.982	0.987	0.984	0.989	0.986	0.992	0.987	0.996	0.990
20	0.873	0.983	0.941	0.989	0.961	0.991	0.979	0.993	0.982	0.993	0.991	0.995	0.992	0.995
25	0.874	0.985	0.935	0.982	0.959	0.992	0.975	0.991	0.983	0.994	0.986	0.995	0.990	0.998
30	0.896	0.982	0.948	0.988	0.973	0.990	0.981	0.991	0.988	0.991	0.992	0.995	0.995	0.995
40	0.904	0.988	0.948	0.986	0.975	0.990	0.985	0.989	0.984	0.991	0.989	0.993	0.992	0.993
50	0.908	0.980	0.965	0.981	0.978	0.988	0.987	0.988	0.994	0.993	0.993	0.993	0.996	0.995
75	0.890	0.987	0.948	0.988	0.969	0.991	0.980	0.993	0.980	0.993	0.989	0.993	0.990	0.995
100	0.902	0.988	0.963	0.984	0.974	0.989	0.985	0.989	0.989	0.991	0.990	0.992	0.992	0.993

Figure 3.10: Accuracy of a LP, running from a normalized polytope $\mathbb{P}^S_N$, to classify normalized local and nonlocal correlations originating from a N'-partites system.

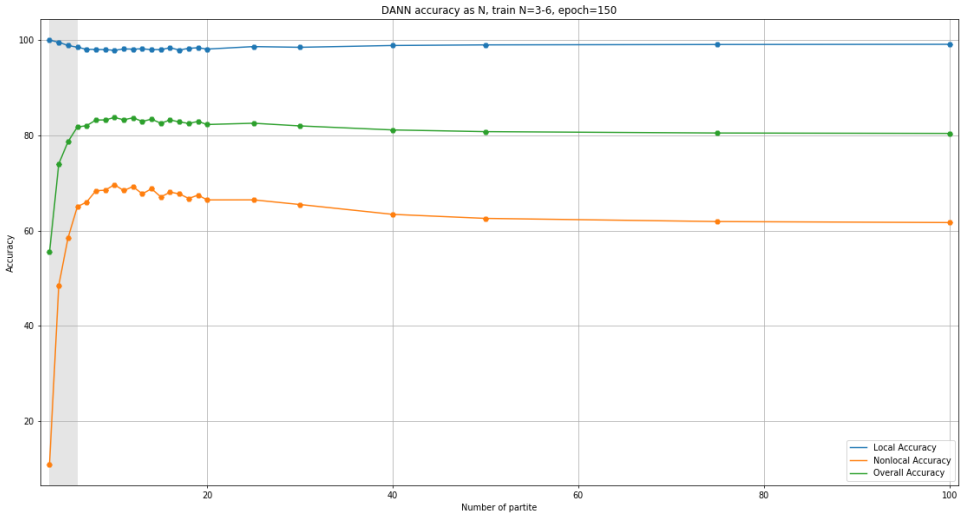

Figure 3.11: DANN accuracy for local and nonlocal correlations. Trained on correlations originating from systems with 3 to 6 parties. Tested on $N \in \{3, 4, 5, \ldots, 21, 25, 30, 40, 50, 75, 100\}$.

Testing time for a given correlation is almost instantaneous – in the order of $10\mu s$.
Using the domain adversarial method resulted in a more stable accuracy than what we achieved with the FNN for both local and nonlocal classification. With the DANN architecture detailed above, we were able to achieve an accuracy of more than 0.9785 for local correlations. Furthermore, our DANN provides a good hint on nonlocality detection with an accuracy higher than 0.6 for $N > 6$.

Chapter 4

Conclusions and Future Work

Conclusions – In this thesis we provided a machine-learing scheme for certifying nonlocality of multipartite correlations, using neural networks and body correlators. In the spirit of Machine Learning, our detection method is computationally efficient at the cost of being not perfectly accurate. However, the results obtained show that testing with a small dataset, the classification accuracy is already fairly high. Indeed, false positives (local correlations classified as nonlocal) happen less than 2.15% of the time, and nonlocality is detected in more than 60% of the cases for a all number of parties higher than 6. What is more, we leveraged the recently introduced Domain Adversarial Neural Networks to provide a testing algorithm which is blind (or, independent) to the number of parties composing the system.

While we were working on this project, a method to test nonlocality using machine-learning was published [42]; however, the aim there is quite different from ours. On the one hand, although very high accuracy is reported in the classification experiments done in [42], the training and testing scenarios coincide, that is, same number of parties, inputs and outputs. What is more, these are of a fairly low complexity and so, for example, for the standard CHSH scenario, running the linear program is already efficient (and perfectly accurate). We aimed at designing a tool which can be trained for feasible number of parties and used for sizes of multipartite systems for which no feasible and perfectly accurate method is known.

Future Work – While the results presented suggest that the Domain Adversarial approach allows for a better classification accuracy, fairly comparing the FNN and the DANN approach will imply maximizing both neu-

ral networks performances, especially for local classification, and this will hence require the optimisation of their architectures and parameters. An approach to this technological challenge is to use a modified neuroevolution algorithm[1], with the objective of maximizing local accuracy, stability of this accuracy, and overall accuracy.
Accuracy of the DANNs might also increase by a blending of an ensemble of FNN for the label classification (as it was done in [42]).
Finally, we would also like to explore the capabilities of the DANNs regarding the developing of classifications schemes independent of the number of measurements (possibly for a fixed number of parties).

[1]The code structure of DANN and FNN, we provide, allows the use of such and algorithm, thanks to a dynamic structure: number of layers, neurones per layer, and other parameters are arguments taken during the initialization of the neural network.

Bibliography

[1] A. Einstein, B. Podolsky, and N. Rosen. "Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?" In: *Physical Review* 47.10 (1935), pp. 777–780. DOI: 10.1103/physrev.47.777. URL: https://doi.org/10.1103/physrev.47.777.

[2] J. S. Bell. "On the Einstein Podolsky Rosen paradox". In: *Physics Physique* 1.3 (1964), pp. 195–200. DOI: 10.1103/physicsphysiquefizika.1.195. URL: https://doi.org/10.1103/physicsphysiquefizika.1.195.

[3] Nicolas Brunner et al. "Bell nonlocality". In: *Reviews of Modern Physics* 86.2 (2014), pp. 419–478. DOI: 10.1103/revmodphys.86.419. URL: https://doi.org/10.1103/revmodphys.86.419.

[4] Stefano Pironio et al. "Device-independent quantum key distribution secure against collective attacks". In: *New Journal of Physics* 11.4 (2009), p. 045021. DOI: 10.1088/1367-2630/11/4/045021. URL: https://doi.org/10.1088/1367-2630/11/4/045021.

[5] Sophie Laplante et al. "Robust Bell inequalities from communication complexity". In: *Quantum* 2 (2018), p. 72.

[6] Miguel Herrero-Collantes and Juan Carlos Garcia-Escartin. "Quantum random number generators". In: *Reviews of Modern Physics* 89.1 (2017). DOI: 10.1103/revmodphys.89.015004. URL: https://doi.org/10.1103/revmodphys.89.015004.

[7] Manabendra Nath Bera et al. "Randomness in quantum mechanics: philosophy, physics and technology". In: *Reports on Progress in Physics* 80.12 (2017), p. 124001. DOI: 10.1088/1361-6633/aa8731. URL: https://doi.org/10.1088/1361-6633/aa8731.

[8] A. Shenoy-Hejamadi, A. Pathak, and S. Radhakrishna. "Quantum Cryptography: Key Distribution and Beyond". In: *Quanta* 6.1 (2017), p. 1. DOI: 10.12743/quanta.v6i1.57. URL: https://doi.org/10.12743/quanta.v6i1.57.

[9] Itamar Pitowsky. "Correlation polytopes: Their geometry and complexity". In: *Mathematical Programming* 50.1-3 (1991), pp. 395–414. DOI: 10.1007/bf01594946. URL: https://doi.org/10.1007/bf01594946.

[10] L. Babai, L. Fortnow, and C. Lund. "Non-deterministic exponential time has two-prover interactive protocols". In: *Computational Complexity* 1.1 (1991), pp. 3–40. DOI: 10.1007/bf01200056. URL: https://doi.org/10.1007/bf01200056.

[11] Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell. *Machine Learning*. Springer Berlin Heidelberg, 1983. URL: https://doi.org/10.1007/978-3-662-12405-5.

[12] Giuseppe Carleo and Matthias Troyer. "Solving the quantum many-body problem with artificial neural networks". In: *Science* 355.6325 (2017), pp. 602–606.

[13] Juan Carrasquilla and Roger G. Melko. "Machine learning phases of matter". In: *Nature Physics* 13.5 (2017), pp. 431–434. DOI: 10.1038/nphys4035. URL: https://doi.org/10.1038/nphys4035.

[14] Hiroki Saito and Masaya Kato. "Machine Learning Technique to Find Quantum Many-Body Ground States of Bosons on a Lattice". In: *Journal of the Physical Society of Japan* 87.1 (2018), p. 014001. DOI: 10.7566/jpsj.87.014001. URL: https://doi.org/10.7566/jpsj.87.014001.

[15] J. Tura et al. "Nonlocality in many-body quantum systems detected with two-body correlators". In: *Annals of Physics* 362 (2015), pp. 370–423. DOI: 10.1016/j.aop.2015.07.021. URL: https://doi.org/10.1016/j.aop.2015.07.021.

[16] Yaroslav Ganin et al. "Domain-Adversarial Training of Neural Networks". In: *Journal of Machine Learning Research* 17.59 (2016), pp. 1–35. URL: http://jmlr.org/papers/v17/15-239.html.

[17] Sandu Popescu and Daniel Rohrlich. "Quantum nonlocality as an axiom". In: *Foundations of Physics* 24.3 (1994), pp. 379–385. DOI: 10.1007/bf02058098. URL: https://doi.org/10.1007/bf02058098.

[18] Stefano Pironio. "Aspects of Quantum Non-Locality". PhD thesis. Université Libre de Bruxelles, (2004). URL: http://quic.ulb.ac.be/_media/publications/2004-thesis-stefano.pdf.

[19] Stefano Pironio. "Lifting Bell inequalities". In: *Journal of Mathematical Physics* 46.6 (2005), p. 062112. DOI: 10.1063/1.1928727. URL: https://doi.org/10.1063/1.1928727.

[20] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization, With Corrections 2008*. Cambridge University Press, 2004. ISBN: 0521833787.

[21] George Svetlichny. "Distinguishing three-body from two-body nonseparability by a Bell-type inequality". In: *Physical Review D* 35.10 (1987), pp. 3066–3069. DOI: 10.1103/physrevd.35.3066. URL: https://doi.org/10.1103/physrevd.35.3066.

[22] Daniel Collins et al. "Bell-Type Inequalities to Detect Truen-Body Nonseparability". In: *Physical Review Letters* 88.17 (2002). DOI: 10.1103/physrevlett.88.170405. URL: https://doi.org/10.1103/physrevlett.88.170405.

[23] Joseph Bowles et al. "Genuinely Multipartite Entangled Quantum States with Fully Local Hidden Variable Models and Hidden Multipartite Nonlocality". In: *Physical Review Letters* 116.13 (2016). DOI: 10.1103/physrevlett.116.130401. URL: https://doi.org/10.1103/physrevlett.116.130401.

[24] Jean-Daniel Bancal et al. "Definitions of multipartite nonlocality". In: *Physical Review A* 88.1 (2013). DOI: 10.1103/physreva.88.014102. URL: https://doi.org/10.1103/physreva.88.014102.

[25] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[26] David Avis et al. *Deriving Tight Bell Inequalities for 2 Parties with Many 2-valued Observables from Facets of Cut Polytopes*. 2004. eprint: arXiv:quant-ph/0404014.

[27] A. L. Samuel. "Some Studies in Machine Learning Using the Game of Checkers". In: *IBM Journal of Research and Development* 3.3 (1959), pp. 210–229. DOI: 10.1147/rd.33.0210. URL: https://doi.org/10.1147/rd.33.0210.

[28] F. Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain." In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: 10.1037/h0042519. URL: https://doi.org/10.1037/h0042519.

[29] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, 1969. ISBN: 0262130432. URL: https://mitpress.mit.edu/books/perceptrons.

[30] Haim Sompolinsky. *Lecture notes, "Introduction: The Perceptron"*. 2013. URL: http://web.mit.edu/course/other/i2course/www/vision_and_learning/perceptron_notes.pdf.

[31] Claudio Rocha, Fernando Akune, and Ahmed El-Shafie. *Artificial Intelligence and Hybrid Systems*. CreateSpace Independent Publishing Platform, 2013. ISBN: 1477554734. URL: https://www.iconceptpress.com/book/artificial-intelligence-and-hybrid-systems/11000040/1202000315/.

[32] Bernard Widrow and Tedd Hoff. "Adaptive "Adaline" neuron using chemical "memistors"". In: *Number Technical Report 1553-2* (1960).

[33] P.J. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Harvard University, 1975. URL: https://books.google.es/books?id=z81XmgEACAAJ.

[34] Michael A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2018. URL: http://neuralnetworksanddeeplearning.com/.

[35] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2016. eprint: arXiv:1609.04747.

[36] Shai Ben-David et al. "A theory of learning from different domains". In: *Machine Learning* 79.1-2 (2009), pp. 151–175. DOI: 10.1007/s10994-009-5152-4. URL: https://doi.org/10.1007/s10994-009-5152-4.

[37] Santosh Vempala. "Geometric random walks: a survey". In: *Combinatorial and Computational Geometry* (2005), pp. 573–612.

[38] Ravi Kannan, Laszlo Lovasz, and Miklos Simonovits. "Random walks and $\mathcal{O}(n^5)$ volume algorithm for convex bodies". In: *Random Structures and Algorithms* 11.1 (1997), pp. 1–50.

[39] R. H. Dicke. "Coherence in Spontaneous Radiation Processes". In: *Physical Review* 93.1 (1954), pp. 99–110. DOI: 10.1103/physrev.93.99. URL: https://doi.org/10.1103/physrev.93.99.

[40] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2011. ISBN: 0387310738.

[41] Pieter-Tjerk de Boer et al. “A Tutorial on the Cross-Entropy Method”. In: *Annals of Operations Research* 134.1 (2005), pp. 19–67. DOI: 10.1007/s10479-005-5724-z. URL: https://doi.org/10.1007/s10479-005-5724-z.

[42] Askery Canabarro, Samuraí Brito, and Rafael Chaves. *Machine learning non-local correlations*. 2018. eprint: arXiv:1808.07069.