Study on estimating quantum discord by neural network with prior knowledge

Yong-Lei Liu · An-Min Wang · Yi Sun · Peng-Fei Zhang · Guo-Dong Wang

Received: 3 February 2019 / Accepted: 30 September 2019 / Published online: 10 October 2019
© Springer Science+Business Media, LLC, part of Springer Nature 2019

Abstract
Machine learning has achieved success in many areas because of its powerful fitting ability, so we hope it can help us to solve some significant physical quantitative problems, such as quantum correlation. In this research, we will use neural networks to predict the value of quantum discord. Quantum discord is a measure of quantum correlation which is defined as the difference between quantum mutual information and classical correlation for a bipartite system. Since the definition contains an optimization term, it makes analytically solving hard. For some special cases and small systems, such as two-qubit systems and some X-states, the explicit solutions have been calculated. However, for general cases, we still know very little. Therefore, we study the feasibility of estimating quantum discord by machine learning method on two-qubit systems. In order to get an interpretable and high-performance model, we modify the ordinary neural network by introducing some prior knowledge which comes from the analysis about quantum discord. Our results show that prior knowledge actually improves the performance of neural network.

Keywords Quantum discord · Machine learning · Neural network

1 Introduction

In recent years, the field of machine learning (ML) has developed rapidly with deep learning, such as the famous Alpha GO [1,2] and various technologies based on
computer vision, such as health care [3], face recognition [4,5]-based identity authentication system, action recognition [6,7]. This method is also used to solve another problem beyond computer science, such as phase transition in condensed matter [8–11], effective representation of quantum multibody state [12], molecular nature prediction [13–15], and so on. Thus, we hope this method will help us to deal with some quantum information problems. In quantum information area, distinguishing separable and entangled states is an important problem. There has been some research on recognizing a bipartite state entangled or separable by machine learning method [16–18]. Thus, the problem that how to measure entanglement is more worth studying, especially for quantum correlation of mixed states. Our work attempts to predict a measure of quantum correlation.

In 2000, Olliver and Zurek [19,20] produced quantum discord (QD) by analogy of classical mutual information to define this measure of quantum correlation. Because of the optimization problem in the definition of QD, it is NP-hard problem to get analytical results for general cases [21]. So, we investigate how to apply ML technology to estimate this measure of quantum correlation. In ML, given a set of data $D$ from a function $f : x \mapsto y$, ML algorithm $A$ can find an approximate function $h$ from a hypothesis set $H$, by training based on the observed data. The function $f$ is called target function. We hope to find an universal method to estimate the QD well, and this method can give a prediction with acceptable error on any bipartite system. In theory, deep neural network (DNN) can fit any continuous target function [22]; however, extracting specific information from such model is hard due to the black-box property of DNN [23,24]. The black box means that it is hard to understand how a DNN gives the result from input step-by-step for human. Thus, we must modify the model to make extracting information easy by introducing prior knowledge based on the analysis about QD.

In Sect. 2, we will introduce the quantum discord of two-qubit systems. In Sect. 3, we will describe the neural network models we used. In Sect. 4, we will show results in two different cases. Finally, we discuss some problems and future works in Sect. 5.

2 QD of two-qubit systems

QD is a measure of quantum correlation mainly for mixed states. In a bipartite quantum system, there are classical correlation and quantum correlation. Assume $\rho^{AB}$ is the density operator of the bipartite system and $\rho^A$ and $\rho^B$ are density operators of its two parts A(lice) and B(ob). By analogy classical mutual information, Olliver and Zurek gave the definition of quantum mutual information as [19]

$$I(\rho^{AB}) = S(\rho^A) + S(\rho^B) - S(\rho^{AB})$$

where $S(\rho) = -\text{tr}(\rho \log_2 \rho) = -\sum_i \lambda_i \log_2 \lambda_i$ is von Neumann entropy and $\lambda_i$ is eigenvalue of corresponding density operator. Let $C(\rho^{AB})$ be classical correlation and $Q(\rho^{AB})$ be quantum correlation. So we can get a kind of quantumness correlation by subtracting classical correlation from quantum mutual information as per the definition [19],

$$Q(\rho^{AB}) = I(\rho^{AB}) - C(\rho^{AB}).$$
This quantum part has been called QD.

Since classical correlation depends on measurement, we used to view part A by measuring part B. Give a set of basis \( \{ \Pi_k | \Pi_k = I \otimes B_k \} \) to measure B, the state of the system after measuring is

\[
\rho_k = \Pi_k \rho^{AB} \Pi_k / p_k
\]

where \( p_k = \text{tr}_B(\Pi_k \rho^{AB}) \) is the probability of measurement [19]. Then, the conditional entropy of the final state is

\[
S(\rho^{AB}|\{B_k\}) = \sum_k p_k S(\rho_k).
\]

Now, the mutual information of this system is [25]

\[
I(\rho^{AB}|\{B_k\}) \equiv S(\rho^A) - S(\rho^{AB}|\{B_k\}).
\]

Henderson and Vedral [26,27] gave the explicitly form of classical correlation

\[
C(\rho^{AB}) = \sup_{\{B_k\}} I(\rho^{AB}|\{B_k\})
\]

\[
= S(\rho^A) - \min_{\{B_k\}} S(\rho^{AB}|\{B_k\}).
\]

From the above, we can know that the quantum correlation of a bipartite system, which is named as QD by Olliver and Zurek [19], is

\[
Q(\rho^{AB}) = S(\rho^B) - S(\rho^{AB}) + \min_{\{B_i\}} S(\rho^{AB}|\{B_i\})
\]

There are a few states that can be solved analytically. In 2008, Luo [28,29] got analytical solution of QD for a kind of qubit–qubit system state which is \( \rho = I + \sum_{j=1}^{3} c_j \sigma_j \otimes \sigma_j \). In 2010, Ali with coworkers [25] got a more general solution of a kind of state which is called X-state whose elements are all zeros except main diagonal and antidiagonal by similar method. In 2015, Maldonado-Trapp et.al. gave some supplements to the results of two-qubit X-states QD [30]. However, it is still a hard task to solve the third term in Eq. (7) for large systems or general cases.

In order to estimate QD by neural network (NN) well, we should choose an appropriate target function to reduce model complexity. In other words, the simpler the target function is, the better performance we can get. For Eq. (7), we can see that the first two terms, \( S(\rho^B) \) and \( S(\rho^{AB}) \), are easy to get analytically solution by solving eigenvalues of \( \rho^B \) and \( \rho^{AB} \), so we just pay attention to the third term of Eq. (7), called optimization term. For convenience, we write it as

\[
c(\rho^{AB}) \equiv \min_{\{B_i\}} S(\rho^{AB}|\{B_i\}).
\]
Table 1  Dimension of the feature space $\Phi$ with highest power $L$, and size of the original parameter space $n$

| $L$ | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  |
|-----|----|----|----|----|----|----|----|----|----|
| $n = 7$ | 8  | 36 | 120| 330| 792| 1716| 3432| 6435| 11,440 |
| $n = 9$ | 10 | 55 | 220| 715| 2002| 5005| 11,440| 24,310| 48,620 |

The first line is the different value of $L$. The second line is the dimension of $\Phi$ when $n = 7$ with different values of $L$ for Eq. (9). The third line is the dimension of $\Phi$ when $n = 9$ with different values of $L$ for Eq. (9).

3 Machine learning model

The original parameters of a two-qubit state are too few to reveal the feature directly. Therefore, before feeding data to a network, we need a feature transformation as preprocessing to make the feature more obvious. We apply a power series transformation by the following rule. Let $n$ be the dimension of the original feature space and $L$ be the highest power after transformation. So, for a vector $x \in \mathbb{R}^n$, the result is $\Phi(x) = (1, x_1, x_2, \ldots, \phi(x)_{a_1 \cdots a_n}, \ldots, x_n^L)^T$, and the element $\phi(x)_{a_1 \cdots a_n}$ is

$$\phi(x)_{a_1 \cdots a_n} = \prod_{i=1}^{n} x_i^{a_i}, \quad \sum_{i} a_i \leq L, \quad (9)$$

where $a_i$ is the power of the $i$th base of $x$. And, we list some size of $\Phi$ for some cases in Table 1.

3.1 Estimating by ordinary NN

Firstly, we use an ordinary NN attempt to estimate QD. The NN model we used has only one hidden layer, see Fig. 1. Assume we have a raw dataset $D = \{(x^{(i)}, c^{(i)}) | i = 1, \ldots, M\}$, where $M$ is the size of the dataset. The input of the NN can be written as a matrix, $X$, formed by feature vectors, and $X_{i,j} = \Phi(x^{(i)})_j$ is the $i$th data’s $j$th feature. The hidden layer $h$ has $F$ nodes. The values of optimization term, or labels for ML, can be written as a vector $y$, where $y_i = c^{(i)}$, which is the label of the $i$th data. $W_1$ and $W_2$ are weight matrices of net connections, $\tilde{y}$ is the predictive value of the model and $\sigma$ is the activation function , $\sigma(x) = 1/(1 + \exp(-x))$. The computations of this NN are

$$h = \sigma(W_1 X)$$
$$\tilde{y} = W_2 h$$

3.2 NN with prior knowledge

Adding prior knowledge about special problem in the model can help us improve the performance of the model. From the definition of QD, we can see that knowledge about entropy is an important concept in this problem. So, our model should know...
Fig. 1  An ordinary NN with one hidden layer. $X$ is the input dataset, $h$ is a hidden layer with $F$ nodes and $\hat{y}$ is the output of this model. $6$ means summing the input for each node, and $\sigma$ is sigmoid activation function. $W_1$ and $W_2$ are the weight matrices between layers.

Fig. 2  A NN model with knowledge about entropy. $X$ is the input dataset, $h$ is a hidden layer with $F$ nodes and $\hat{y}$ is the output of this model. $6$ means summing the input for each node, and $E$ is knowledge about entropy defined as Eq. (11). $W_1$ and $W_2$ are the weight matrices between layers.

something about entropy. From Eq. (4), we can know that $c$ is a sum of von Neumann entropy of a density operator ensemble, $\{p_i, \tilde{\rho}_i\}$, and the only problem is that we do not have an effective method to find it. Thus, we are sure that getting $\{p_i, \tilde{\rho}_i\}$ is equivalent to getting their eigenvalues, so we have the following form

$$c = \sum_{i,j} p_i \tilde{\lambda}_{i,j} \log_2 \tilde{\lambda}_{i,j}$$  \hspace{1cm} (10)

where $\tilde{\lambda}_{i,j}$ are eigenvalues of density operator $\tilde{\rho}_i$. Thus, we replace the activation function by a new “activation function”

$$E(x) \equiv \begin{cases} 0, & x \leq 0; \\ x \log_2 x, & x > 0. \end{cases}$$  \hspace{1cm} (11)

In this NN model, the input of this layer may be the approximate of eigenvalues, ideally. What’s more, the change of activation function can further reduce the model complexity for previous layer.
We illustrate this model in Fig. 2, and we call it PKNN for convenience. The computations of this model are,

\[
    h = E(W_1 X) \\
    \hat{y} = W_2 h.
\]

The second layer realizes the sum in the definition in von Neumann entropy in some way. And, we do not need offset term in this layer according to the definition of entropy.

### 3.3 Double branch NN with prior knowledge

We should note one thing that because of the optimizer, \( \min_{B_i} \), the result of QD is shattered. In other words, there are some conditions, \( g(x) \geq 0 \), that cause the target function \( c(x) \) or eigenvalue \( \tilde{\lambda}_{i,j} \) to have different analytical forms in different parameter ranges,

\[
    c(x) = \begin{cases} 
    c_1(x), & g_1(x) \geq 0; \\
    \vdots & \vdots \\
    c_n(x), & g_n(x) > 0.
    \end{cases} \tag{12}
\]

The above can be written in one formula

\[
    c(x) = \sum_i \theta(g_i(x)) \ c_i(x) \tag{13}
\]

here, \( \theta(\cdot) \) is step function

\[
    \theta(a) = \begin{cases} 
    0, & a \leq 0; \\
    1, & a > 0.
    \end{cases} \tag{14}
\]

Unfortunately, how much conditions are there in the target function is unknown in advance; therefore, we should choose an appropriate value in an experiment.

Based on the above discussion, we modified the model again, the new one named double branch neural network (DBNN), see Fig. 3. In this model, we add condition control factors in a new path. Ideally, the value of a condition function \( g \), or output of \( h_c \), should belong to \( \{0, 1\} \). However, step function is hard for training, so we choose sigmoid function, \( \sigma \), as alternative. The whole computations of this model are

\[
    h = E(W_1 X) \\
    h_c = \sigma(W_{\text{cond}} X) \\
    (y_p)_{i,j} = h_{i,j}(h_c)_{i,j} \\
    \hat{y} = W_2 y_p
\]

The layers of \( h \) and \( h_c \) are used to probe eigenvalues \( \tilde{\lambda}_{i,j} \) and condition functions \( g(x) \); then, they are an approximation of eigenvalues and conditions.
Fig. 3 A double branch NN with conditional function. $X$ is the input dataset. $h$ is a hidden layer with $F$ nodes for eigenvalues and $h_c$ is a hidden layer with $F$ nodes too for condition function. $\tilde{y}$ is output of this model. $y_p$ is middle layer by multiplying $h$ and $h_c$. $l_6$ means summation of input, $\sigma$ is sigmoid activation function and $E$ is knowledge about entropy. $W_1$, $W_2$ and $W_{\text{cond}}$ are the weight matrices between layers.

4 Result

We will compare the above three models in this section. To evaluate the performance of these models, we use mean quadratic loss function as index:

$$\ell = \frac{1}{M} \sum_{i} (y^{(i)} - \tilde{y}^{(i)})^2.$$  \hspace{1cm} (15)

At last, these models are trained by TensorFlow [31].

4.1 Result on X-state

First, we test our models on two-qubit X-states,

$$\rho_{X}^{AB} = \begin{pmatrix} \rho_{11} & 0 & 0 & \rho_{14} \\ 0 & \rho_{22} & \rho_{23} & 0 \\ 0 & \rho_{32} & \rho_{33} & 0 \\ \rho_{41} & 0 & 0 & \rho_{44} \end{pmatrix},$$  \hspace{1cm} (16)
Table 2 Results of the three models on X-state

|       | NN     | PKNN   | DBNN   |
|-------|--------|--------|--------|
|       | Training ($\times 10^{-3}$) | Test ($\times 10^{-3}$) | Training ($\times 10^{-3}$) | Test ($\times 10^{-3}$) |
| 3.433 | 3.162  | 1.427  | 1.690  | 1.052  | 1.163  |
| 3.435 | 3.167  | 1.071  | 1.157  | 1.069  | 1.266  |
| 3.007 | 2.887  | 1.340  | 1.352  | 1.211  | 1.319  |
| 2.994 | 2.875  | 1.513  | 1.691  | 0.946  | 1.040  |
| 3.007 | 2.839  | 1.475  | 1.443  | 0.991  | 1.114  |
| $\bar{\ell}$ | 2.986  | 1.365  | 1.467  | 1.054  | 1.180  |

The five-row data in the middle of this table are the five times result of these three models. The last line is average of loss $\bar{\ell}$. BDNN has the best performance, and PKNN is second and we can use a seven-dimension real vector, $\mathbf{x} \in \mathbb{R}^7$, to characterize it. Thus, we have

$$\rho_{11} = x_1, \rho_{22} = x_2, \rho_{33} = x_3, \rho_{44} = 1 - x_1 - x_2 - x_3, \rho_{14} = x_4 + i x_5, \rho_{23} = x_6 + i x_7.$$

According to the eigenvalues of $\rho^{AB}_X$ [25], there are some extra restrictions for those parameters:

$$|\rho_{14}|^2 \leq \rho_{11}\rho_{44},$$

$$|\rho_{23}|^2 \leq \rho_{22}\rho_{33}.$$ (17)

So, for X-state, the optimization term is a function of a 7-D vector:

$$c(\rho_X) \in \mathcal{H}_X(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^7$$

In this experiment, we use a training set with $M = 6000$ to inhibit over-fitting. These data are numerical solutions coming from randomly generated two-qubit X-states. In order to balance direct feature and the amount of calculation, we choose $L = 6$ as the highest power of transformation and the second layer has $F = 16$ nodes. Every model runs five times and runs $3 \times 10^5$ steps each time. Then, we compare their average performance. $3 \times 10^5$ steps can guarantee that loss function will not change obviously in every turn. Other training parameters are the same; they are initial learning rate $\eta = 0.2$ and decay with 0.98 every 3000 steps.

Table 2 shows all results, and the last line is the average value of loss function, $\bar{\ell}$. We can see that PKNN is better than an ordinary NN and DBNN has the best performance on average. This accords with our expectation.

It is hard to find best result each time because of random initialization. One of the reasons is that there are lots of local minima in those models. Moreover, the nonmonotonic function $E(\cdot)$ makes the loss function more complex than monotonic activation function. For the same reasons, those models are unstable. Even so, these results can prove that prior knowledge can improve the performance of NN.

Then, we show the performance of the best result in Table 2, and the result is plotted in Fig. 4. The horizontal axis is the theoretical value of optimization term or label, $c$, and the vertical axis is the predictive value, $\tilde{c}$, from DBNN. We can see that the predictive value falls in a bit wide range.
Fig. 4  Best result of DBNN on X-state. $c$ is theoretical value, and $\tilde{c}$ is predictive value. a The performance on training set, b the performance on test set. The blue line in a and b is $\tilde{c} = c$. c Error between $\tilde{c}$ is predictive value and $c$ is theoretical value on training set. d Error between $\tilde{c}$ is predictive value and $c$ is theoretical value on test set (Color figure online)

Next, we test our model on a X-state

$\rho = \begin{pmatrix} 0.35 - 0.35a & 0 & 0 & -0.2 + 0.2a \\ 0 & 0.25 + 0.25a & -0.15 + 0.6a & 0 \\ -0.2 + 0.2a & 0 & 0 & 0.2 - 0.2a \end{pmatrix}$. (18)

It is easy to get the analytical solution by Ali’s result [25],

$c(\rho) = \min S(\rho|\{B_i\}) = \min \{S'(\rho)|_{\theta_1, \theta_2}, S'(\rho_0)|_{\theta_3}, S'(\rho_0)|_{\theta_4}\}$ (19)

where

$S'(\rho)|_{\theta_j} = -\frac{1 - \theta_j}{2} \log_2 \frac{1 - \theta_j}{2} - \frac{1 + \theta_j}{2} \log_2 \frac{1 + \theta_j}{2}$ (20)

$S'(\rho)|_{\theta_1, \theta_2} = (0.55 + 0.15a)S'(\rho_0)|_{\theta_1} + (0.45 - 0.15a)S'(\rho_1)|_{\theta_2}$ (21)
Fig. 5 Analytical solution and prediction of a DBNN. The red solid line is Eq. (19), and the three cyan dashed lines are $S'(\rho)_{\theta_1,\theta_2}$, $S'(\rho_0)_{\theta_3}$ and $S'(\rho_0)_{\theta_4}$. The blue ‘×’ points are our prediction (Color figure online)

and

$$
\theta_1 = \frac{\sqrt{(0.15 - 0.85 a)^2}}{0.55 + 0.15 a} \\
\theta_2 = \frac{\sqrt{(0.05 + 0.25 a)^2}}{0.45 - 0.15 a} \\
\theta_3 = \frac{\sqrt{0.1325 - 0.62 a + 0.73 a^2}}{0.5} \\
\theta_3 = \frac{\sqrt{0.0125 - 0.02 a + 0.25 a^2}}{0.5}
$$

(22)

Although we can extract formula from this model like

$$
\sum_{i=1}^{L} (W_2)_i \ E((W_1 X)_i) \frac{1}{1 + \exp((-W_c X)_i)}
$$

it is difficult to matching with Eq. 19 due to two reasons. First, the precision of model is not enough; second, the form of a expression is not unique. So, we plot the analytical solution and the predictive result of a DBNN in Fig. 5. It shows that we have a good prediction about QD including condition.

4.2 Result on real state

To show the universality of our method, we test it on a special two-qubit state which is researched little, and we call it real state. Its elements are all real, $\rho_{ij} \in \mathbb{R}$. We can use nine independent parameters to characterize it:

$$
\rho^{AB}_R = \begin{pmatrix}
\rho_{11} & \rho_{12} & \rho_{13} & \rho_{14} \\
\rho_{21} & \rho_{22} & \rho_{23} & \rho_{24} \\
\rho_{31} & \rho_{32} & \rho_{33} & \rho_{34} \\
\rho_{41} & \rho_{42} & \rho_{43} & \rho_{44}
\end{pmatrix} = \begin{pmatrix}
x_1 & x_4 & x_5 & x_6 \\
x_4 & x_2 & x_7 & x_8 \\
x_5 & x_7 & x_3 & x_9 \\
x_6 & x_8 & x_9 & x_0
\end{pmatrix}, \ x_0 = 1 - x_1 - x_2 - x_3.
$$

(23)
Table 3 Results of ordinary NN and DBNN on real state

|                | NN Training ($\times 10^{-3}$) | NN Test ($\times 10^{-3}$) | DBNN Training ($\times 10^{-3}$) | DBNN Test ($\times 10^{-3}$) |
|----------------|--------------------------------|-----------------------------|---------------------------------|-------------------------------|
|                | 2.200                          | 1.996                       | 1.104                           | 1.183                         |
|                | 2.193                          | 1.981                       | 1.099                           | 1.168                         |
|                | 2.195                          | 1.984                       | 1.047                           | 1.178                         |
|                | 2.196                          | 1.992                       | 1.162                           | 1.237                         |
|                | 2.216                          | 2.007                       | 1.228                           | 1.270                         |
| $\bar{\ell}$  | 2.200                          | 1.992                       | 1.128                           | 1.207                         |

The five-row data in the middle of this table are the five times result of these three models. The last line is average $\bar{\ell}$. DBNN is better than ordinary NN model.

The solution of this state is unknown; thus, we do not know the range of nondiagonal elements. So we generate this kind of state randomly with following empirical restrictions to improve sampling efficiency

$$|\rho_{ij}|^2 \leq \rho_{ii}\rho_{jj}$$  \hspace{1cm} (24)

and discard the one with negative eigenvalue. Well, for a real state, the optimization term is a function of a nine-dimensional vector:

$$c(\rho_R^{AB}) \in \mathcal{H}_R(x), \quad x \in \mathbb{R}^9$$

The truncated term is still at $L = 6$. From previous calculations in Table 1, we know that the size of the first layer is 5005, which is huge. The size of the training set is the same as before, $M = 6000$. The initial learning rate $\eta = 0.2$ and decay with 0.96 every 3000 steps in this experiment. As a transitional model, we do not test PKNN this time.

Table 3 shows all results, and Fig. 6 displays the best result of DBNN in Table 3. We can see that DBNN has similar loss with X-state on real states training set and test set. The range of data points in Fig. 6 is also similar to Fig. 4. By comparing the results of ordinary NN and DBNN, we can see that our modification based on prior knowledge actually improves the performance of the network.

5 Discussion

Before testing those models, we try to use general DNN that has five hidden layers with 32 nodes and sigmoid activation function but does not have feature transformation in Sect. 3 to estimate QD; however, the result is very bad. It cannot fit QD. The reason we think is that there is complex interaction between parameters of density operator. What's more, if the general DNN would successfully fit QD, we do not know how to explain such model and we do not know what is the meaning of sigmoid activation function for QD. That’s why we construct above three models.
In principle, neural network can fit any continuous target function [22]. We have a simple viewpoint. For any continuous function, \( f(x) : x \mapsto \mathbb{R}, \ x \in \mathbb{R}^n \). Let \( d(x_1, x_2) \) be the Euclidean distance of these two points. At last, let \( \delta(z) \) be a fast decay function with absolute value of \( z \), such as \( \exp(-z^2) \) or \( 1/(1+z^2)^n \), \( n \in \mathbb{R}^+ \). Now, if we have a dataset \( D = \{ (x^{(m)}, y^{(m)} = f(x^{(m)})) | m = 1, 2, \ldots, M \} \), the following model will give a good fit on it:

\[
o(x) = \sum_{m=1}^{M} w_m y^{(m)} \delta(\eta_m (x - x^{(m)}))
\]

(25)

where \( w_m \) and \( \eta_m \) are called model parameters to training. Adjacent \( n + 1 \) points can make a good prediction for a small area surrounded by them due to continuity. However, such model is huge, but simplifying it is difficult. And, such model does not help achieve our goal, that is, to get an interpretable model or to get a model which can extract solution form. This is another reason we build new model.

DNN gives us some inspiration. For example, for polynomial equation, we can build a special network to express its solution, such as the solution of \( ax^2 + bx + c = 0 \)

\[
x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}
\]
and we can express it as a network-like model, see Fig. 7. We call it PW (power and weight) network.

The output of an unit in P(ower)-layer is

$$\text{out}_i = \prod_{j=1}^{\text{in}_j} (\text{in}_j)^{p_{ij}}, \quad p_{ij} \in \mathbb{R}.$$  \hfill (26)

The output of an unit in W(eight)-layer is

$$\text{out}_i = \sum_{j=1}^{\text{in}_j} w_{ij} \text{in}_j, \quad w_{ij} \in \mathbb{R}.$$  \hfill (27)

in$_j$ are input of this unit, and output of an unit in W(eight)-layer is.

6 Conclusion

In summary, we build two new models by adding prior knowledge about entropy which are based on the analysis about the problem in this work. Results show that prior knowledge assuredly improves the performance of NN. What’s more, our model has an advantage which is the interpretability that builds model according to the analysis about problem.

Acknowledgements This work is supported by Key Research and Development Plan of Ministry of Science and Technology, China (No. 2018YFB1601402).
References

1. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driesche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., Hassabis, D.: Mastering the game of Go with deep neural networks and tree search. Nature 529, 484 (2016)

2. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driesche, G., Graepel, T., Hassabis, D.: Mastering the game of Go without human knowledge. Nature 550, 354 (2017)

3. Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., Cui, C., Corrado, G., Thrun, S., Dean, J.: A guide to deep learning in healthcare. Nat. Med. 25, 24–29 (2019)

4. Sáez Trigueros, D., Meng, L., Hartnett, M.: Face Recognition: From Traditional to Deep Learning Methods. arXiv e-prints arXiv:1811.00116 (2018)

5. Wang, M., Deng, W.: Deep Face Recognition: A Survey. arXiv e-prints arXiv:1804.06655 (2018)

6. Zhu, W., Hu, J., Sun, G., Cao, X., Qiao, Y.: A key volume mining deep framework for action recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016), pp. 1991–1999. https://doi.org/10.1109/CVPR.2016.219

7. Yan, S., Xiong, Y., Lin, D.: Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. arXiv e-prints arXiv:1801.07455 (2018)

8. van Nieuwenburg, E.P.L., Liu, Y.H., Huber, S.D.: Learning phase transitions by confusion. Nat. Phys. 13, 435 (2017)

9. Zhang, Y., Kim, E.A.: Quantum loop topography for machine learning. Phys. Rev. Lett. 118, 216401 (2017)

10. Ch'ng, K., Carraquilla, J., Melko, R.G., Khatri, T.E.: Machine learning phases of strongly correlated fermions. Phys. Rev. X 7, 031038 (2017)

11. Broecker, P., Carraquilla, J., Melko, R.G., Trebst, S.: Machine learning quantum phases of matter beyond the fermion sign problem. Sci. Rep. 7, 8823 (2017)

12. Gao, X., Duan, L.M.: Efficient representation of quantum many-body states with deep neural networks. Nat. Commun. 8, 662 (2017)

13. Rupp, M., Tkatchenko, A., Muller, K.R., Von Lilienfeld, O.A.: Fast and accurate modeling of molecular atomization energies with machine learning. Phys. Rev. Lett. 108, 058301 (2012)

14. Montavon, G., Rupp, M., Gobre, V., Vazquez-Mayagoitia, A., Hansen, K., Tkatchenko, A., Muller, K.R., Von Lilienfeld, O.A.: Machine learning of molecular electronic properties in chemical compound space. New J. Phys. 15(9), 095003 (2013)

15. Hansen, K., Montavon, G., Biegler, F., Fazli, S., Rupp, M., Scheffler, M., Von Lilienfeld, O.A., Tkatchenko, A., Muller, K.R.: Assessment and validation of machine learning methods for predicting molecular atomization energies. J. Chem. Theor. Comput. 9(8), 3404 (2013)

16. Ma, Y.C., Yang, M.H.: Transforming Bell’s inequalities into state classifiers with machine learning. npj Quantum Inf. 4, 34 (2018)

17. Lu, S., Huang, S., Li, K., Li, J., Chen, J., Lu, D., Ji, Z., Shen, Y., Zhou, D., Zeng, B.: Separability-entanglement classifier via machine learning. Phys. Rev. A 98, 012315 (2018)

18. Gao, J., Qiao, L.F., Jiao, Z.Q., Ma, Y.C., Hu, C.Q., Ren, R.J., Yang, A.L., Tang, H., Yung, M.H., Jin, X.M.: Experimental machine learning of quantum states. Phys. Rev. Lett. 120, 240501 (2018)

19. Ollivier, H., Zurek, W.H.: Quantum discord: a measure of the quantumness of correlations. Phys. Rev. Lett. 88, 017901 (2001)

20. Zurek, W.H.: Decoherence, einselection, and the quantum origins of the classical. Rev. Mod. Phys. 75, 715 (2003)

21. Huang, Y.: Computing quantum discord is NP-complete. New J. Phys. 16, 033027 (2014)

22. Cybenko, G.: Approximation by superpositions of a sigmoidal function. Math. Control Signals Syst. 2(4), 303 (1989)

23. Mordvintsev, A., Ohah, C., Tyka, M.: Inceptionism: going deeper into neural networks (2015). https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

24. Reimann, J.N., Schwung, A.: Neural logic rule layers (2019). https://doi.org/10.13140/RG.2.2.10091.59687

25. Ali, M., Rau, A.R.P., Alber, G.: Quantum discord for two-qubit X states. Phys. Rev. A 81, 042105 (2010)

26. Henderson, L., Vedral, V.: Classical, quantum and total correlations. J. Phys. A 34, 6899 (2001)
27. Vedral, V.: Classical correlations and entanglement in quantum measurements. Phys. Rev. Lett. 90, 050401 (2003)
28. Li, N., Luo, S.: Total versus quantum correlations in quantum states. Phys. Rev. A 76, 032327 (2007)
29. Shun-Long, L.: Quantum discord for two-qubit systems. Phys. Rev. A 77, 042303 (2008)
30. Maldonado-Trapp, A., Hu, A., Roa, L.: Analytical solutions and criteria for the quantum discord of two-qubit X-states. Quantum Inf. Process. 14(6), 1947 (2015)
31. Tensorflow. https://www.tensorflow.org

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.