1. The comparisons with state-of-art quantum neuron models

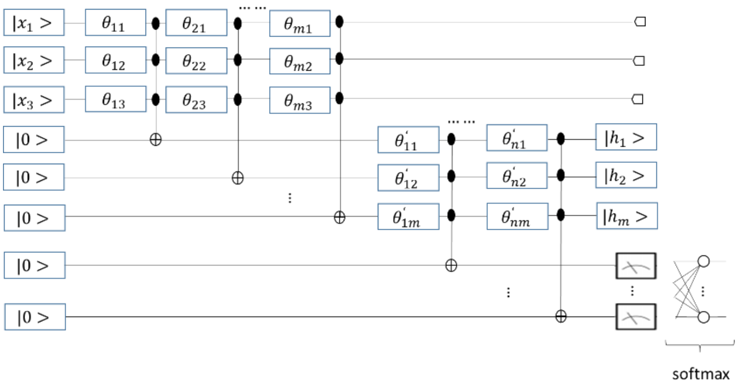

Fig1. State-of-art quantum neuron models

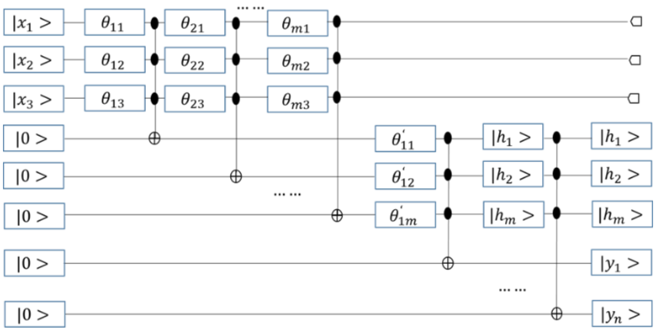

Fig2. The quantum neuron models of this work

Compared with state-of-art quantum neural networks, we have two structural innovations. First, the state-of-art quantum neural networks always connect a softmax layer in the end, so the number of qubits in the output layer must be same as the number of classifications, thus the multi-classification tasks (more than 50) cannot be realized. Our network cancels this restriction, because the eigenstate corresponds a classification label in our model that will be discussed in the second part. Second, we can get the classification result by quantum measurement directly, every eigenstate corresponds a classification label, and we can get the classification result by one measure. But in state-of-art quantum neural networks, they need multiple measurements to get probability distributions.

2. The learning algorithm for this network should be described in more detail

First of all, we need define the loss function,

$$error = \sum_{i=0}^{c} (\tilde{y}_k - y_k)^2$$

$\tilde{y}_k$ is the measurement result, y_k is the label of this sample, note that the $\tilde{y}_k$(or y_k) is the eigenstate, such as 10010...1, 11001...0, 00110...0, etc. that we can calculate the gradient, the θ'_{km} and θ_{mn} is the parameter in our network,

$$\frac{\partial error}{\partial \theta'_{km}} = \frac{\partial error}{\partial y_k} \times \frac{\partial y_k}{\partial \theta'_{km}} = (y_k - \mathrm{y}) \times \prod_{m} sin\,(\arcsin(h_m) + \theta'_{km})\,/\tan((\arcsin(h_m) + \theta'_{km})$$

$$\frac{\partial error}{\partial \theta_{mn}} = \frac{\partial error}{\partial y_k} \times \frac{\partial y_k}{\partial h_m} \times \frac{\partial h_m}{\partial \theta_{mn}}$$

$$= (y_k - \mathrm{y}) \times \frac{\prod_m sin\,(\arcsin(h_m) + \theta'_{km})}{\tan((\arcsin(h_m) + \theta'_{km})}/(1 - h_m^2)$$

$$\times \prod_{n=3} sin\,(\arcsin(x_n) + \theta_{mn})\,/\tan(\arcsin(x_n) + \theta_{mn})$$

after get the gradient, we can implement the gradient descent

$$\theta'_{km}(t+1) = \theta'_{km}(t) - \eta \frac{\partial error}{\partial \theta'_{km}}$$

$$\theta_{mn}(t+1) = \theta_{mn}(t) - \eta \frac{\partial error}{\partial \theta_{mn}}$$

t is the number of iteration steps and η is the learning rate. The training will end until the results converge.