The Neural Network Variational Monte Carlo library

Jan Kessler   Francesco Calcavecchia

July 12, 2018

1 FFNNWaveFunction

In case you want to use a Neural Network as Wave Function, there is an already prepared WaveFunction. One needs to provide only the number of spacial dimensions (`nspacedim`) and number of particles (`npart`), and a FeedFowardNeuralNetwork (see the library) that has `nspacedim` $\times$ `npart` inputs and only one output. This FFNN should be already connected, but should not have any derivative substrate. As for the `WaveFunction`, some flags can be specified in the constructor to tell whether the variational derivatives are computed or not. Internally, this class create two separated instances of this FFNN, a bare one, which does not have any substrate, and is used only for sampling, and a more complex one, that is used for computing all the derivatives.

Of course, it inherits all the methods from the WaveFunction class, and we do not report it in the following.

```
class FFNNWaveFunction: public WaveFunction{

private:
    FeedForwardNeuralNetwork * _ffnn;

public:
    // --- Constructor
    // IMPORTANT: The provided ffnn should be ready to use (
        connected) and have the first, second and variational
        derivatives substrates
    FFNNWaveFunction(const int &nspacedim, const int &npart,
        FeedForwardNeuralNetwork * ffnn, bool flag_vd1=true, bool
         flag_d1vd1=true, bool flag_d2vd1=true);


    // --- Getters
    FeedForwardNeuralNetwork * getBareFFNN(){return _bare_ffnn;}
    FeedForwardNeuralNetwork * getDerivFFNN(){return _deriv_ffnn
        ;}

};
```