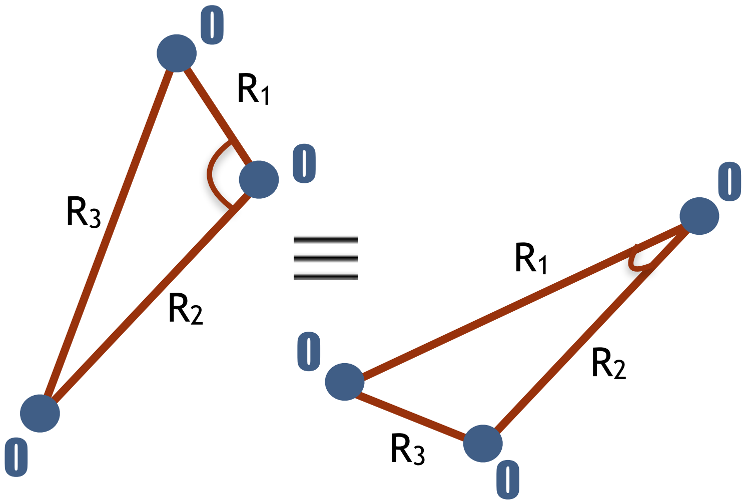
ANN for PESs: Methodology

Multi-layer feed-forward Neural Networks (NN) have been adopted **as fitting functional:**

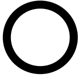


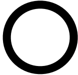
0
R1
0
R3
R2
0
=
0
R1
0
R2
R3
0

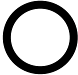
1. A **Symmetrized Polynomial Vector** (G) is constructed, in order to account for the permutation symmetries; for example, for a A3-type system:

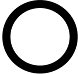
where , being and tunable parameters.

Permutation Invariant Polynomials Neural Networks (PIP-NN):









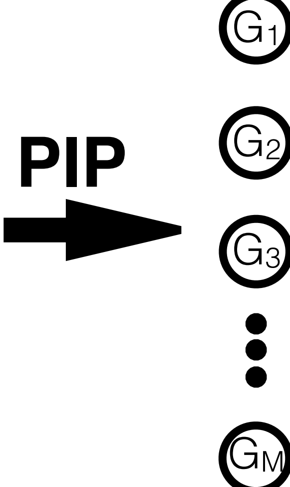
●

R_1

R_2

R_3

R_N


PIP
G_1
G_2
G_3
G_M

- Easy to implement;
- Easy to train;
- Easy to generalize to new systems;
- Easy to differentiate in **R**;

- Cost effective;
- Easy to be refined;
- Widely tested;
- Easy to be extended to the stochastic case.

13

$$G_1 = p_1 + p_2 + p_3$$

$$G_2 = p_1 p_2 + p_2 p_3 + p_1 p_3$$

$$G_3 = p_1 p_2 p_3$$

$$p_i = \exp(-\lambda_i(R_i - r_{e_i}))$$

r_{e_i}

λ_i

theano

Lasagne

ANN for PESs: Methodology

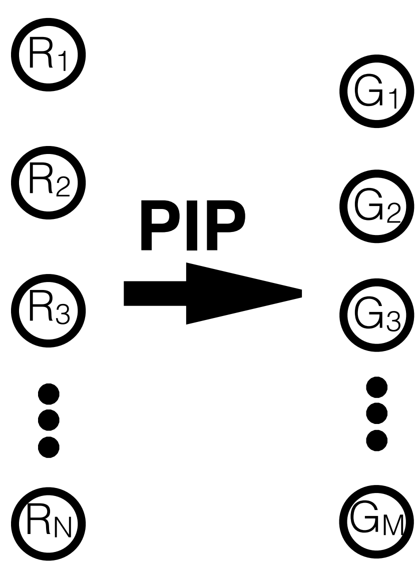
Multi-layer feed-forward Neural Networks (NN) have been adopted **as fitting functional**:

- ✦ Easy to implement;
- ✦ Easy to train;
- ✦ Easy to generalize to new systems;
- ✦ Easy to differentiate in **R**;
- ✦ Cost effective;
- ✦ Easy to be refined;
- ✦ Widely tested;
- ✦ Easy to be extended to the stochastic case.

Permutation Invariant Polynomials Neural Networks (PIP-NN):

theano

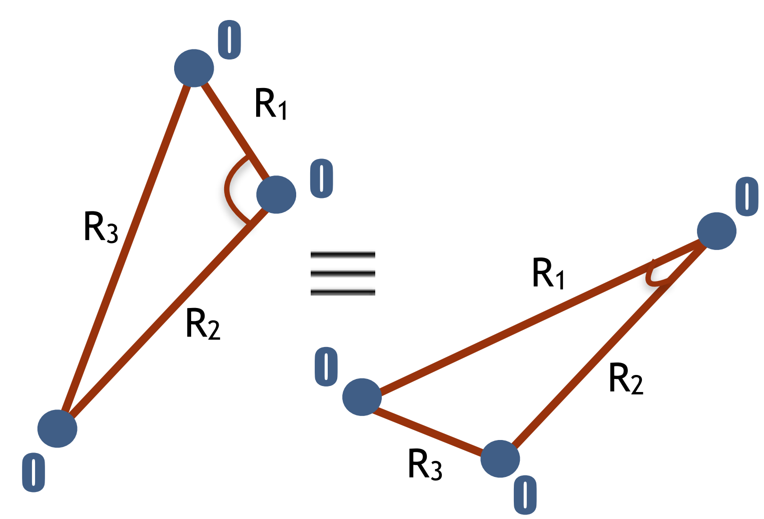
Lasagne



1. A **Symmetrized Polynomial Vector** (**G**) is constructed, in order to account for the permutation symmetries; for example, for a A3-type system:

$$G_1 = p_1 + p_2 + p_3$$
$$G_2 = p_1p_2 + p_2p_3 + p_1p_3$$
$$G_3 = p_1p_2p_3$$

where $p_i = \exp(-\lambda_i(R_i - r_{e_i}))$, being λ_i and r_{e_i} tunable parameters.



ANN for PESs: Methodology

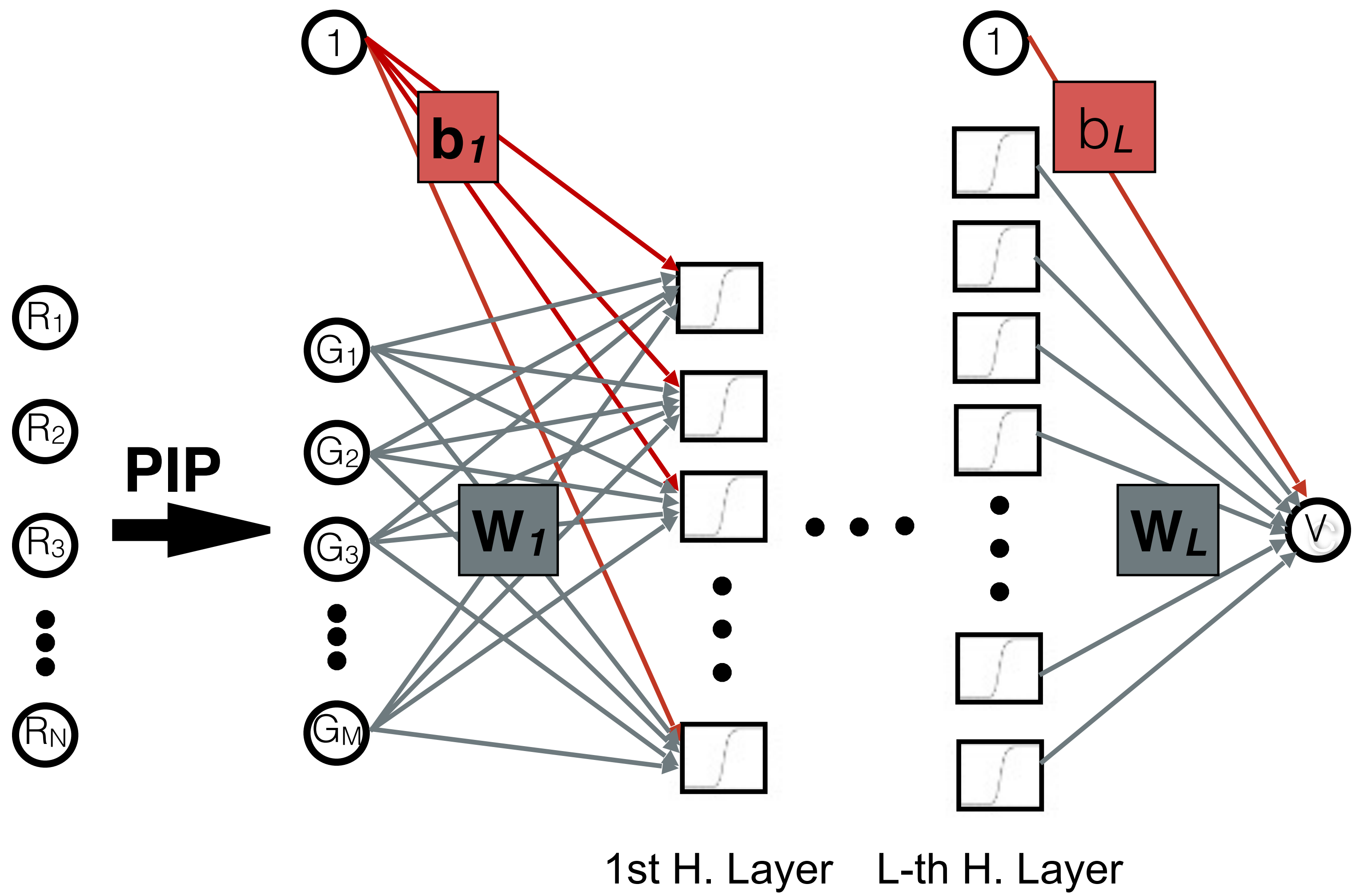
Multi-layer feed-forward Neural Networks (NN) have been adopted **as fitting functional**:

- ✦ Easy to implement;
- ✦ Easy to train;
- ✦ Easy to generalize to new systems;
- ✦ Easy to differentiate in **R**;
- ✦ Cost effective;
- ✦ Easy to be refined;
- ✦ Widely tested;
- ✦ Easy to be extended to the stochastic case.

Permutation Invariant Polynomials Neural Networks (PIP-NN):

theano

Lasagne



2. **G** is fed to a feed-forward neural network, and it **flows through its layers** as a series of weighted linear combinations alternated to non-linear functions

Output from the k-th Neuron of the i-th Layer

$$\begin{cases} z_i^k = \sum_{j=1}^{N_{i-1}} W_i^{jk} y_{i-1}^j + b_i^k \\ y_i^k = f_i\left(z_i^k\right) \end{cases}$$