



機械学習と物理化学

理論 & 実践

機械学習？

データサイエンス



ビッグデータ

機械学習

インフォマティクス

具体的にはどんなことをするのか？

データを元に、予測や分類モデルを自動的に
つくる仕組み全般

と言われても具体性に欠けると思うので、
実際に機械学習タスクを動かしてみましよう

一般的な機械学習の導入

<https://github.com/eminamitani/lecture-Minamitani/blob/master/ML.ipynb>

化学分野での話題

https://github.com/eminamitani/lecture-Minamitani/blob/master/ML_chemistry.ipynb

固体物理での応用：機械学習ポテンシャル

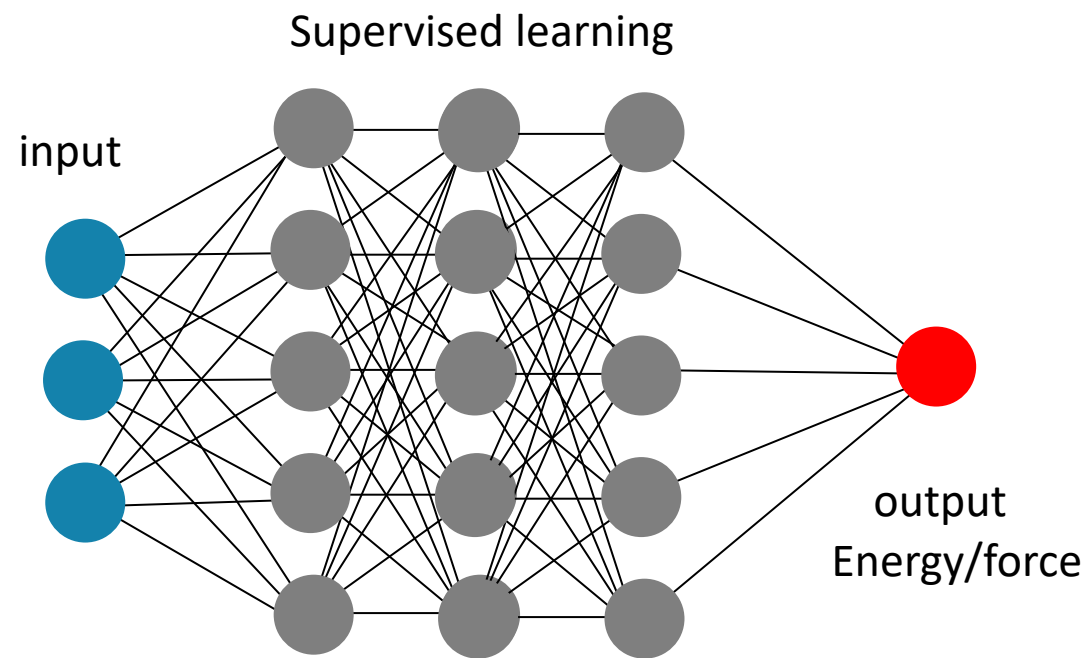
第一原理計算は精度が良いけど重すぎる

第一原理計算がしていること $F: R^N \rightarrow R(\text{energy}), R^3(\text{force})$

構造をインプットにしてエネルギーや力を返す

この機能を機械学習モデルで代用できないか？

ポテンシャルを機械学習で作ろう



経験的ポテンシャルとの違い

経験的ポテンシャル

物理的な要請から決めた関数形がある

原子-原子間の相互作用でだいたい決まるだろう

Embedded Atom Method (EAM) ポテンシャル

$$E_{tot} = \sum_i E_i(n_i) + \frac{1}{2} \sum_{i \neq j} \phi_{ij}(r_{ij})$$

各原子に割り当てられたエネルギー

距離に依存する関数
 $\exp \left[-b^l \left(\frac{r_{ij}}{r_e} - 1 \right) \right]$

などを使うことが多い

ボンド間の角度も重要なのでは？

Tersoff ポテンシャル

$$V = \frac{1}{2} \sum_i \sum_{i \neq j} f_c(r_{ij}) [F_R(r_{ij}) + b_{ij} F_A(r_{ij})]$$

$$b_{ij} = (1 + \beta_i^n \zeta_{ij}^n)^{-1/2n} \quad \zeta_{ij}^n = \sum_{k \neq i, j} f_c(r_{ik}) g(\theta_{ijk})$$

3つの原子の座標で決まる角度に依存
→多体ポテンシャル

経験的ポテンシャルとの違い

経験的ポテンシャル：意味合いもわかりやすい&解析微分式が得られる→力の計算、MDでの利点

- ・ **パラメータ最適化の労力がとてつもない**
- ・ **パラメータや関数形が固定されてしまう**
→特定の環境に合わせて最適化したパラメータを、他の系で使っていていいのかは微妙

Ex) sp^2/sp^3 のように結合にバリエーションがある炭素原子

ダイヤモンド構造は Tersoffポテンシャルでも再現できるが、
そのポテンシャルだと グラフェン・カーボンナノチューブでは微妙...

より柔軟性が高く、最適化にも人力を使わなくて済むのが
機械学習ポテンシャルのメリット



機械学習ポテンシャルの種類

ポテンシャルの関数形は規定しない

Gaussian Approximation Potential (GAP)

A. P. Bartok et al., Phys. Rev. Lett. 104, 136403 (2010)

カーネル法がベース

(High-dimensional) Neural Network potential (HDNNP)

J. Behler and M. Parrinello, Phys. Rev. Lett. 98, 146101 (2007)

ニューラルネットワーク

ポテンシャルの関数形はある程度決める

Physically informed artificial neural network

EAMや多体ポテンシャルのパラメータを
ニューラルネットワークにする

G. P. Purja Pun et al., Nat. Commun. 10, 2339 (2019)

他の新しいアーキテクチャ：
グラフニューラルネットワークベース

SchNet

K. T. Schutt et al., Arxiv: 1706.08566 (2017)

PFP

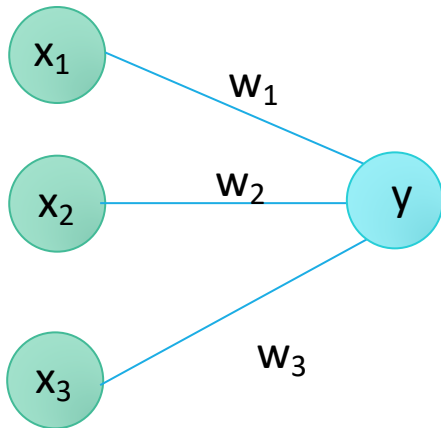
S. Takemoto et al., Nat. Commun. 21,2991 (2022)
など

ニューラルネットワーク

ニューラルネットワークの構成要素：パーセプトロン

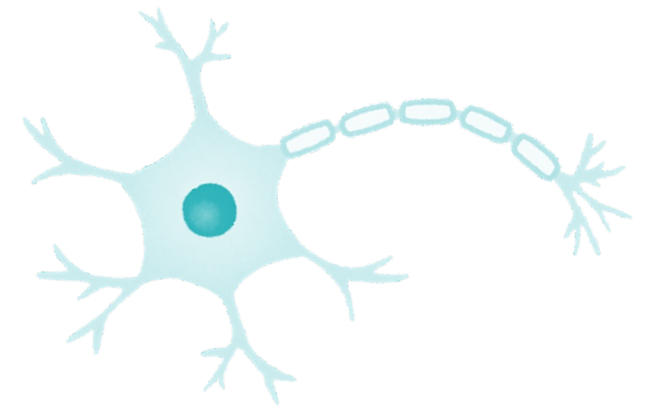


ニューロンの発火モデル

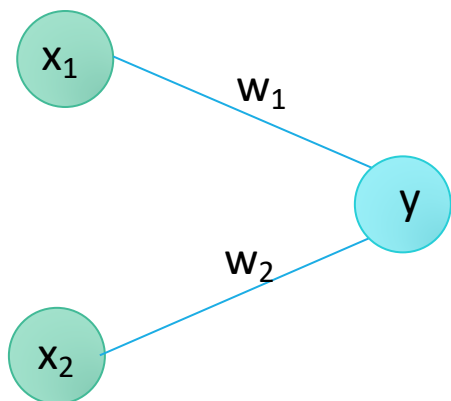


$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 + w_3x_3 \leq \theta) \\ 1 & (w_1x_1 + w_2x_2 + w_3x_3 > \theta) \end{cases}$$

入力に重みをかけたものの総和が
しきい値より大きい小さいかで、0または1を返す



パーセプトロンでできること



ANDゲートの動作

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

$$w_1=0.3, w_2=0.6, \theta=0.8$$

$$w_1x_1 + w_2x_2$$

$$0$$

$$0.6 < \theta$$

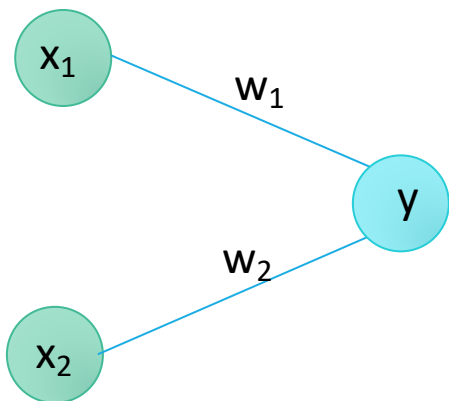
$$0.3 < \theta$$

$$0.9 > \theta$$

他にも様々な組み合わせで実現できる

パーセプトロンでできること

重みやしきい値を変えるだけでOR, NANDゲートの動作にもできる



OR

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

Ex) $w_1=0.3$, $w_2=0.6$, $\theta=0.2$

NAND

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

Ex) $w_1=-0.3$, $w_2=-0.2$, $\theta=-0.4$

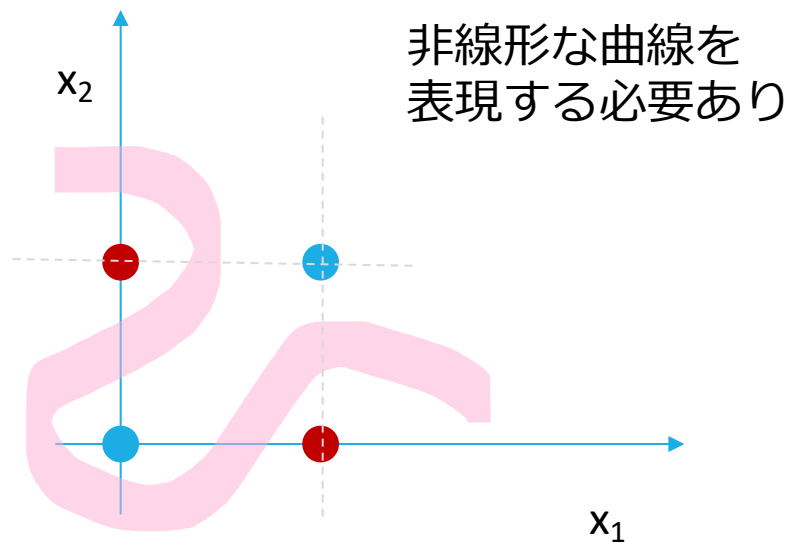
パーセプトロンでできること

XORゲートは作れるか？

XOR

x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

XOR→線形分離できない

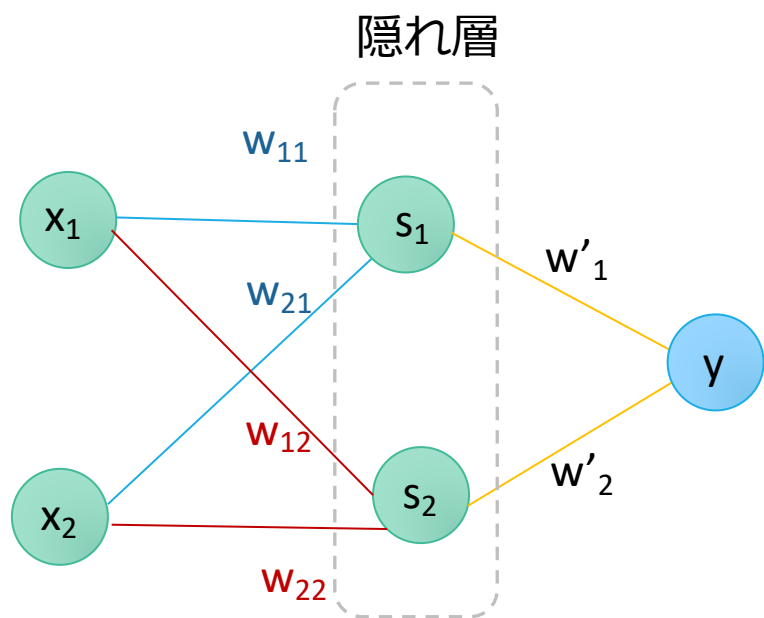


パーセプトロン1つでは無理

↓
重ねると実現できる

↓
多層パーセプトロン

多層パーセプトロン



考えられる例

青い部分($w_{11}=-0.5$, $w_{21}=-0.5$, $\theta=0$)

赤い部分($w_{12}=0.5$, $w_{22}=0.5$, $\theta=0.8$)

黄色い部分 ($w'_1=0.4$, $w'_2=0.6$, $\theta=0.3$)

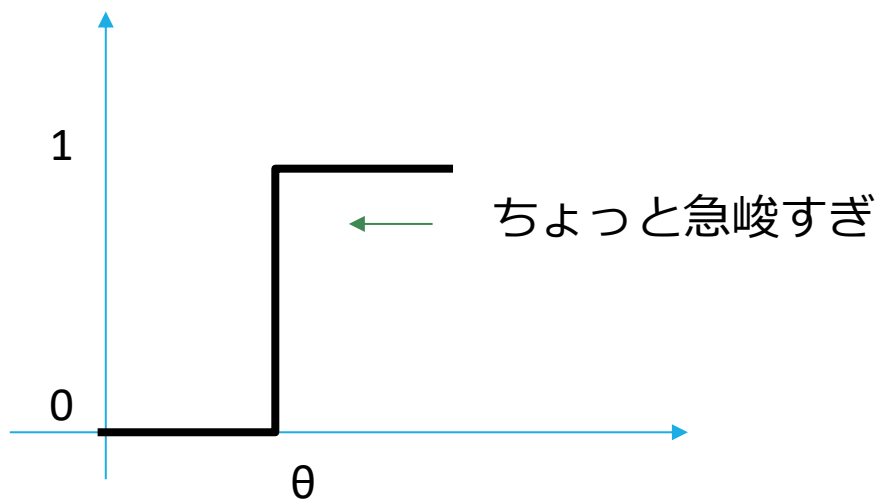
x_1	x_2	s_1	s_2	y
0	0	1	0	1
0	1	0	0	0
1	0	0	0	0
1	1	0	1	1

パーセプトロンを多層にすることで非線形な関数も表現できる
→ 万能近似定理

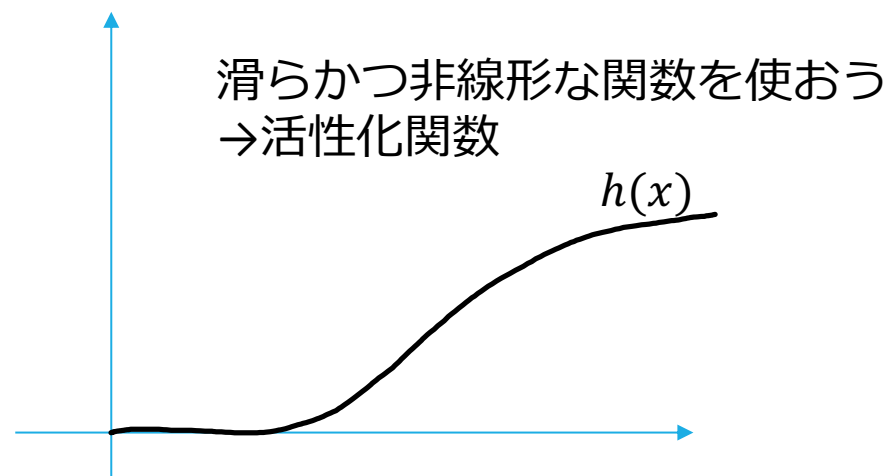
(隠れ層が一つあればいくつかの条件のもと、
任意の連続関数を近似できる)

活性化関数

パーセプトロンではある閾値より上か下かで 0, 1 を切り替えていた



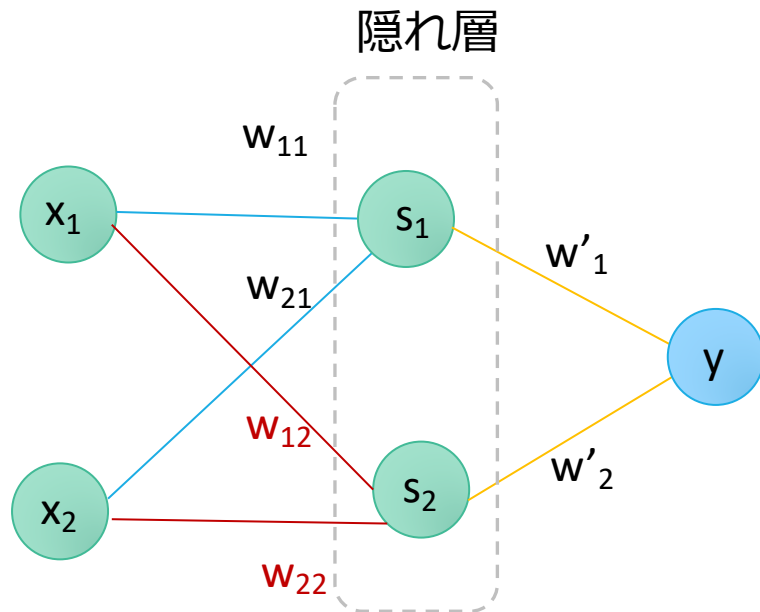
$$y = \begin{cases} 0 & (w_1x_1 + w_2x_2 \leq \theta) \\ 1 & (w_1x_1 + w_2x_2 > \theta) \end{cases}$$



$$y = h(w_1x_1 + w_2x_2)$$

ニューラルネットワークと機械学習へ

インプットと出力の関係を実現する重み・閾値（バイアス）は無限にある



活性化関数を滑らかにしておくことで、微分を使った誤差逆伝播法で重み・バイアスを最適化できる

あるパラメータのネットワークから得られた出力と、真の値で決まる誤差：損失関数の計算結果

$Loss(y, y_{true})$ e.g. 最小二乗誤差

その偏微分をつかってパラメータを更新する

$$w_{11} = w_{11} - \eta \frac{\delta(Loss(y, y_{true}))}{\delta w_{11}} \quad \text{勾配法}$$

これを繰り返すことでよい近似ネットワークを得る→機械学習

機械学習「ポテンシャル」にするには？

構造からエネルギーを予測するモデル

構造のインプットがデカルト座標そのままというのはあまり良くない

物理的要請

系の持つエネルギーは、空間並進・回転や同種原子の入れ替えでも不変

原子の持つ座標をそのまま入力してしまうと、この対称性を学習させることが難しい

学習モデルとしての要請

原子数が異なる系に対応できるモデルである必要がある

原子座標をベクトルデータに埋め込む方法

smooth overlap atomic positions (SOAP)

1. 原子座標を中心としたガウス関数を設定
2. 同径分布関数 + 球面調和関数の線形結合で記述し直すと、
原子の重なりは

線形結合の係数

$$p(r_i)_{nn'l}^{Z_1 Z_2} = \pi \sqrt{\frac{8}{2l+1}} \sum_m c_{nlm}^{Z_1}(r_i)^* \boxed{c_{n'lm}^{Z_2}(r_i)} \quad \text{のモーメント} \quad (p(r_i)_{nn'l}^{Z_1 Z_2} \cdot p(r_j)_{nn'l}^{Z_1 Z_2}) \quad \text{で表現できる}$$

S. De et al., Phys. Chem. Chem. Phys. 18, 13745-13769 (2016)

A. P. Bartók, R. Kondor and G. Csányi, Phys. Rev. B, 87, 184115 (2013)

原子座標をベクトルデータに埋め込む方法

Atom centered symmetry function (ACSF)

J. Behler, J. Chem. Phys. 134, 074106 (2011)

2つの原子対の距離の分布

$$G_i^2 = \sum_{j \neq i} e^{-\eta(R_{ij}-R_s)^2} f_c(R_{ij})$$

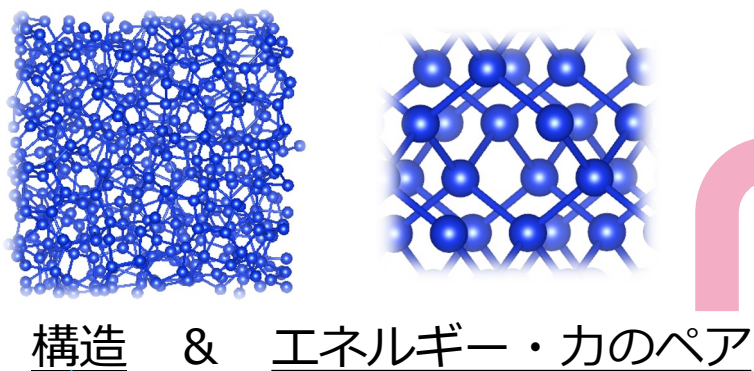
3つの原子が作る角度の分布

$$G_i^4 = 2^{1-\zeta} \sum_{j \neq i, k \neq i, j} (1 + \lambda \cos \theta_{ijk}) e^{-\eta(R_{ij}^2 + R_{ik}^2)} f_c(R_{ij}) \cdot f_c(R_{ik})$$

代表的なこれらの記述子の計算
→オープンソースパッケージでも可能
<https://singroup.github.io/dscribe/latest/>

HDNNPの概要

1. 様々な構造の第一原理計算結果を集める

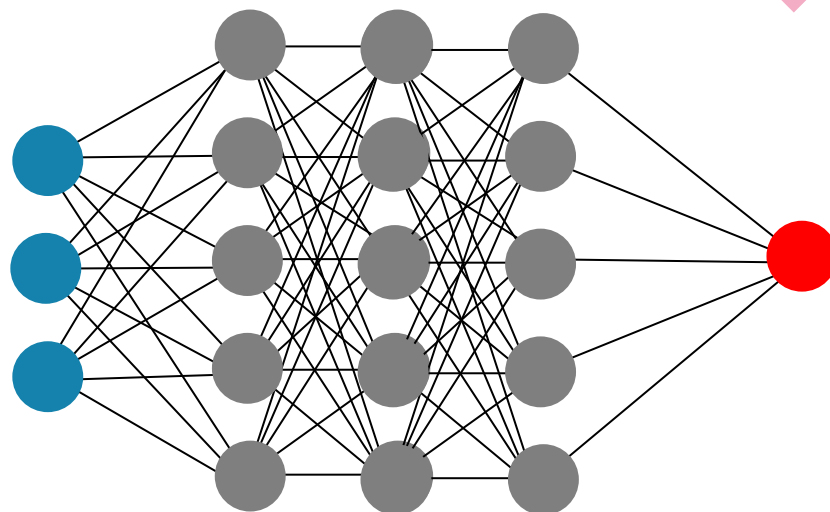


2. 対称性関数などでベクトルデータに加工

$$[G_i^2(\eta_1, R_{s_1}), \dots, G_i^4(\zeta_1, \eta_1, \lambda_1)]$$

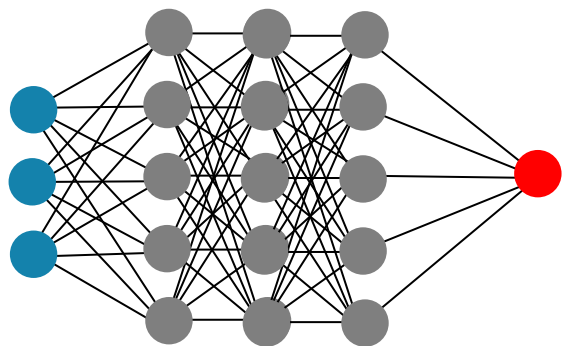
ハイパーパラメータを振ることで
高次元データを作る

3. 機械学習で重みとバイアスを最適化



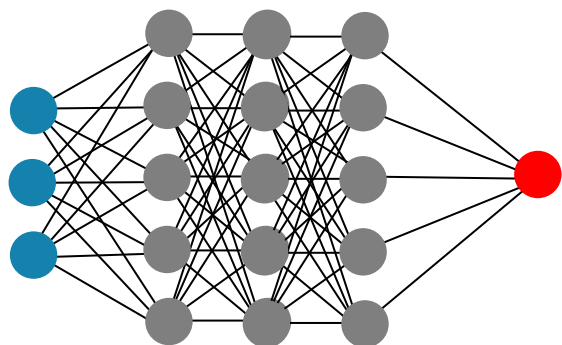
何が高次元なのか

原子 1 の
データ



原子 1 の
エネルギー

原子 2 の
データ



原子 2 の
エネルギー

⋮

⋮

⋮

- ・ ネットワークは同じ種類の元素では同じものを使う
- ・ 全体はネットワークの足し上げで表現

→ 「高次元」

総和→
系の全エネルギー

(この量で損失関数を計算する)

原子数への依存性のなさ
& 同一原子の入れ替えでの不変性を担保

HDNNPのパッケージ

The Atomic Energy Network (**ænet**)

<http://ann.atomistic.net/>

n2p2

<https://compphysvienna.github.io/n2p2/>

Simple-NN

https://github.com/MDIL-SNU/SIMPLE-NN_v2

他にも沢山あるが、ある程度ドキュメンテーションが揃っているものを抜粋。

GAPやSchNetベースも含めるとより選択肢が増える。

GAPの場合はQUIPというパッケージが整備されている。<https://libatoms.github.io/GAP/index.html>

また、VASP6にはGAPの作成機能がある

高度なパッケージは読むのが大変、サクッと概要を眺めてみたい
→ トイモデルを用意してあります

https://github.com/eminamitani/sample_NNP

HDNNP トイモデルを動かしてみる

- 手元にファイルをダウンロードしてくる

`git clone git@github.com:eminamitani/sample_NNP.git` (対称性関数の計算済みデータも含まれている)

- 手元にpython3環境あり→仮想環境 + pytorchインストール
- 手元にpython3環境なし→Google Collaboratoryを使う

自分のGoogle Driveにdesc.npy, label.npyをアップロード

sample_NNP.ipynbかsample_NNP2.ipynb(GPU版)をGoogle Collaboratoryでアップロード

冒頭にGoogle Driveのマウント

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

データの読み込みパスの変更

```
import numpy as np
desc=np.load('/content/drive/MyDrive/Colab Notebooks/data/sample_NNP/desc_large2.npy')
label=np.load('/content/drive/MyDrive/Colab Notebooks/data/sample_NNP/label_large2.npy')
```

この2箇所の変更で動くはずですよ

↑
各自の環境に合わせてください