

Neural Networks

Shyue Ping Ong

University of California, San Diego

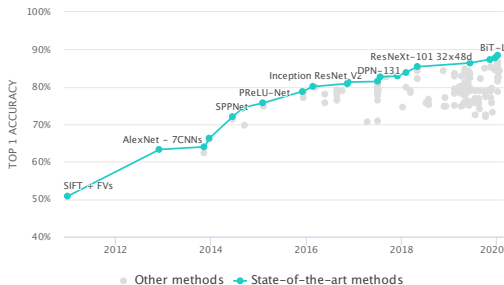
NANO281

Overview

- 1 Preliminaries
- 2 Neural Networks
- 3 Applications

Preliminaries

- Neural networks/deep learning has gotten a lot of hype in recent years.
- In many areas, they have outperformed many traditional ML methodologies.

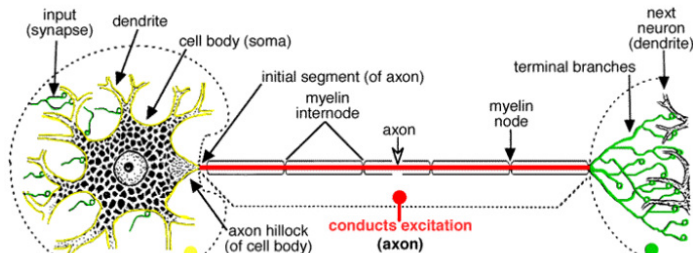


Artificial Neural Network

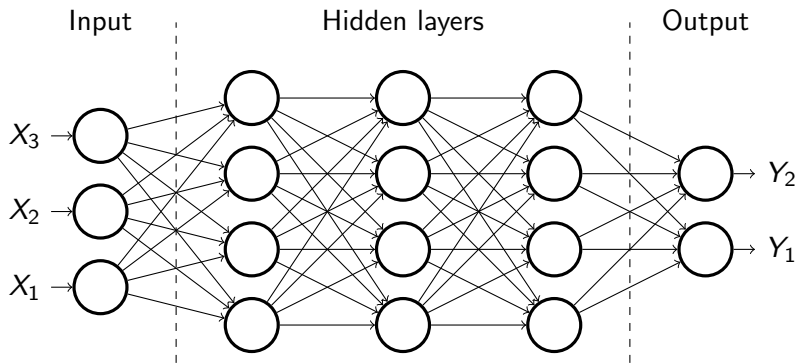
- An artificial neural network (ANN) is a learning algorithm that is (very) loosely based on the structure of the brain.
- Somewhat vaguely, you will also hear the terms “multi-layer perceptron” (MLP) used in place of ANN.

Universal Approximation Theorem^[1]

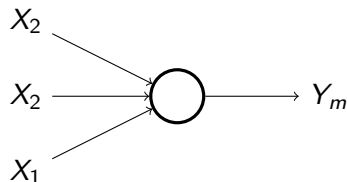
A feed-forward network with a single hidden layer containing a finite number of neurons can approximate continuous functions under mild assumptions on the activation function.



Neural Networks



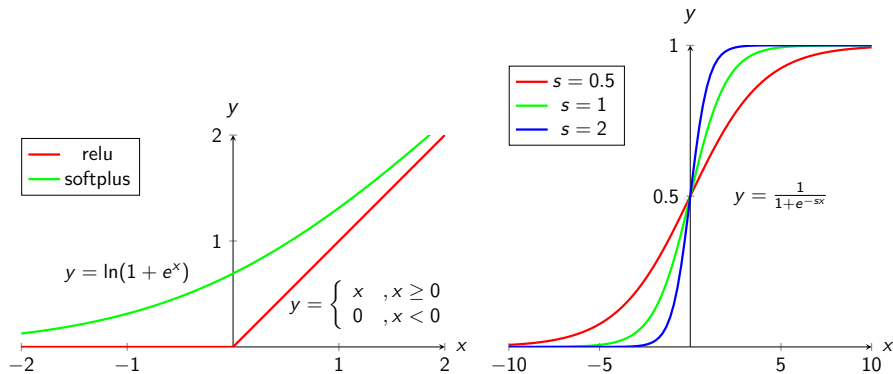
Neuron



$$Y_m = \sigma(\alpha_{0m} + \alpha_m^T \mathbf{X})$$

- Output of each neuron is a linear function of the inputs.
- In hidden layers, the output is passed through an *activation function* σ .
- Common choices of σ include the rectified linear unit (RELU), sigmoid function, softplus, etc.

Activation Functions



Fitting a Neural Network

- Model *weights* (α in the linear functions) are fitted by *back-propagation*, basically a form of gradient descent.
- Loss functions: squared error for regression, squared error or cross entropy for classification.
- To avoid overfitting, regularization (similar to ridge regression) is typically applied. E.g., *weight decay*:

$$J = \sum \alpha^2$$

Parameter Decisions for Neural Networks

- Number of hidden units and layers: generally error on the side of having too many hidden units than too few - flexibility is needed to capture non-linearities in the data.
- Extra weights can be shrunk to zero with appropriate regularization.
- Learning rate is a key parameter in fitting ANNs as well as other models. The learning rate controls the rate of gradient descent. A high learning rate reduces training time, but also decreases accuracy. State-of-the-art is to use an adaptive learning rate.
- The error function is non-convex and contains multiple minima, i.e., final solution obtained depends on initial weights. Typical approach is to start with a number of random starting configurations and choose solution with lowest penalized error.
- Alternative is to average predictions over a number of ANN, i.e., ensemble models.

Software packages for Neural Networks

- Scikit-learn has a basic implementation of a multi-layer perceptron. For this course, we will use this to avoid overloading students with different software packages.
- However, you should be aware that if you are into serious work with ANNs, alternative software are available that offer more features and efficiency. Note that these packages are not limited to ANNs, but ANNs are the most common use case.
 - **Tensorflow**. Open-source package by Google. Very powerful and efficient, but a very steep learning curve. Highly recommend that you use the Keras package (already integrated into TF2) which provides a high level API to it similar to scikit-learn in some ways.
 - **Pytorch**. Open-source package by Facebook. Equally powerful as TF but also steep learning curve.

ANNs in Scikit-learn and TF2.0

```
from sklearn.neural_network import MLPRegressor
```

```
nn = MLPRegressor(hidden_layer_sizes=(5, 3),  
                  alpha=1e-4,  
                  max_iter=200,  
                  learning_rate_init=0.01)
```

```
nn.fit(x, y_reg)
```

```
# Equivalent Tensorflow 2.0
```

```
from tensorflow.keras import Model, Input  
from tensorflow.keras.layers import Dense  
from tensorflow.keras.optimizers import Adam
```

```
x_in = Input(shape=(x.shape[1],))  
x_h1 = Dense(5, activation="relu")(x_in)  
x_h2 = Dense(3, activation="relu")(x_h1)  
x_out = Dense(1)(x_h2)  
model = Model(inputs=x_in, outputs=x_out)  
opt = Adam(learning_rate=0.01)  
model.compile(optimizer=opt, loss='mse')  
model.fit(x, y_reg, epochs=200, batch_size=200)
```

Beyond the Feedforward Neural Network

- ANNs can be an entire course in itself. Here, we cover some basic variations.

Convolutional NNs Uses convolutional operations to achieve translational invariance. Especially important in image processing applications. In modeling crystals with periodic boundary conditions, this can also be a useful feature.

Recurrent NNs Connections between nodes form a directed graph along a temporal sequence. Exhibits temporal dynamic behavior and can handle variable length sequences of inputs. Applications: handwriting recognition or speech recognition.

Example Application: NN for interatomic potentials

- Atom-centered symmetry functions (ACSF)[2] to represent the atomic local environments.

$$G_i^{\text{atom,rad}} = \sum_{j \neq i}^{N_{\text{atom}}} e^{-\eta(R_{ij}-R_s)^2} \cdot f_c(R_{ij}),$$

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

where R_{ij} is the distance between atom i and neighbor j ; η is the width of the Gaussian; R_s is the position shift over all neighboring atoms; ζ controls the angular resolution. $f_c(R_{ij})$ is a cutoff function.

- Fully connected ANNs describe the PES with respect to ACSFs.[3]
- Has been developed for Si, TiO_2 , water, metal-organic frameworks, ZnO , Li_3PO_4 , etc.

Example Application: NNP for Si

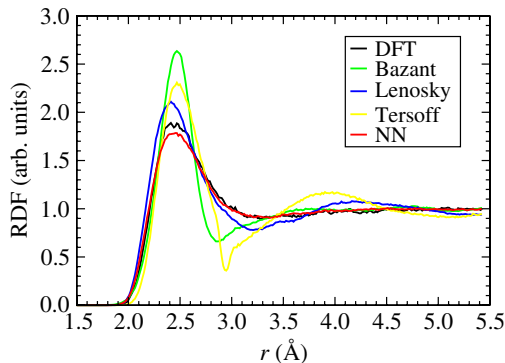
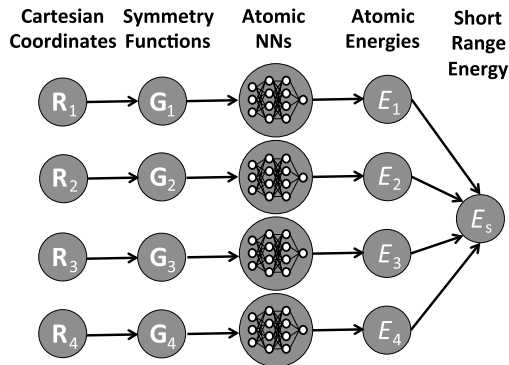
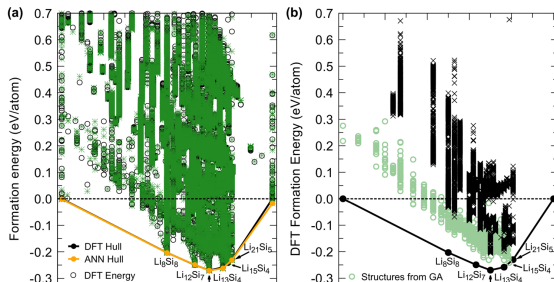
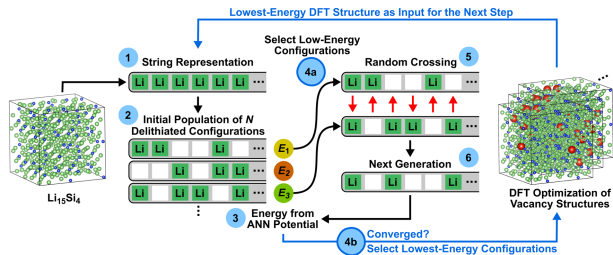
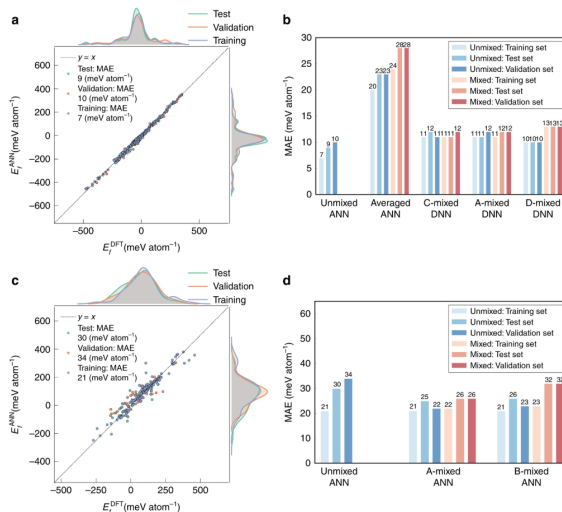
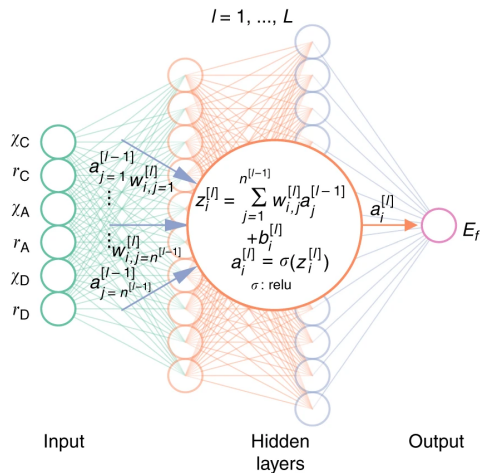


Figure: Left: Schematic of NNP architecture. Right: Comparison of radial distribution functions from MD simulations using various interatomic potentials for Si.[4]

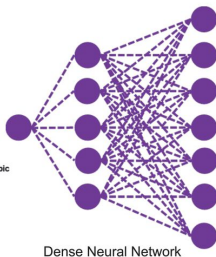
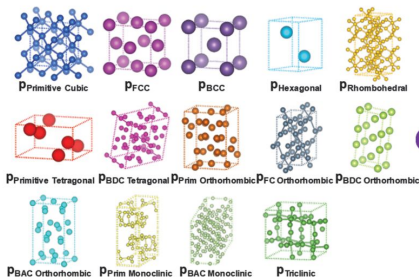
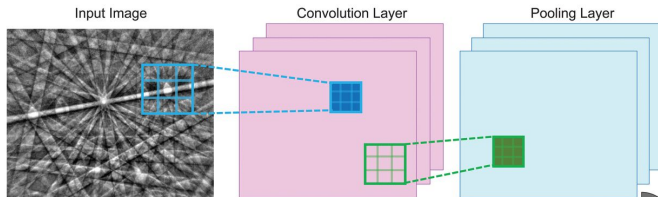
Example Application: NNP-computed phase diagram for amorphous Li-Si



Example application: ANNs to predict crystal stability



Example application: CNNs for determination of crystal symmetry from electron diffraction



A

Patterns Not Used in Training

True class	Primitive Cubic	Face-Centered Cubic	Body-Centered Cubic	Hexagonal	Rhombohedral	Primitive Tetragonal	Body-Centered Tetragonal	Primitive Orthorhombic	Face-Centered Orthorhombic	Body-Centered Orthorhombic	Base-Centered Orthorhombic	Primitive Monoclinic	Base-Centered Monoclinic	Triclinic	
Mo ₅ Si	7297	4	2	5	22	10	0	3	40	2	424	14	12	6176	0.9%
Al	2	8251	7	0	11	1	0	0	80	0	13	88	1	9176	1.0%
Ta	20	58	1440	195	71	75	83	2	1	3	85	14	28	9676	0.1%
Ti	8	3	3	1286	12	7	8	0	3	4	3	3	0	9376	0.0%
Ilmenite	68	3	11	12	4346	189	4	3	75	5	28	32	10	9376	0.0%
Sn	2	3	8	3	0	1842	0	0	0	0	0	28	12	9676	0.0%
Anatase	0	3	2	1	16	0	1262	0	0	0	0	4	0	9376	0.0%
Enstatite	0	0	0	3	3	1	0	3234	0	1	3	0	0	9376	0.0%
Al ₂ Zr ₂	21	8	1	3	11	7	1	1	6788	0	224	21	1	9676	0.0%
MoP ₂	0	2	11	9	34	1	0	1	1	1	1	1	0	9376	0.0%
AuSn ₄	0	0	0	1	0	0	0	0	0	0	0	0	0	9176	0.0%
Malachite	1	0	0	0	1	1	2	1	0	0	1	1	0	9676	0.0%
Fe ₄ Al ₁₃	1	41	7	0	9	9	0	0	0	0	2	18	1045	9476	1.0%
Labradorite	7	0	0	0	7	4	0	0	0	0	0	14	10	3544	0.0%

Predicted class


B


Distinct Material Set

True class	Primitive Cubic	Face-Centered Cubic	Body-Centered Cubic	Hexagonal	Rhombohedral	Primitive Tetragonal	Body-Centered Tetragonal	Primitive Orthorhombic	Face-Centered Orthorhombic	Body-Centered Orthorhombic	Base-Centered Orthorhombic	Primitive Monoclinic	Base-Centered Monoclinic	Triclinic	
V ₂ Cr	7397	54	58	74	43	6	55	16	2	4	16	3	34	0	95.7%
Si	4233	1	12	8	242	5	22	7	404	2	26	0	22	0	83.7%
NbC	0	1642	0	0	1	2	0	0	0	0	0	0	1	0	99.8%
TaC	1	2387	18	14	3	3	3	1	0	0	2	4	1	2	97.9%
Beta Ti	15	43	11919	0	40	222	0	0	0	1	17	24	52	7	96.3%
Fe	0	1	6432	0	0	0	0	0	0	0	0	0	0	3	99.9%
Quartz	0	0	0	0	2052	3	7	0	53	3	0	0	0	3	97.7%
Mo ₂ C	13	7	125	97	17	3	1	2	147	48	593	4281	2	0	92.6%
Dicapside	0	0	0	2	8	1	3	665	0	2	6	2	1	0	0.1%


Predicted class

Bibliography

 Balázs Csanád Csáji.
Approximation with Artificial Neural Networks.
PhD thesis.

 Jörg Behler.
Atom-centered symmetry functions for constructing high-dimensional neural network potentials.
Journal of Chemical Physics, 134(7), 2011.

 Jörg Behler.
High-Dimensional Neural Network Potentials for Complex Systems.
Angewandte Chemie International Edition, pages n/a–n/a.

 Jörg Behler and Michele Parrinello.
Generalized Neural-Network Representation of High-Dimensional Potential-Energy Surfaces.
Physical Review Letters, 98(14):146401, April 2007.

The End