Neural Network Tutorial & Application in Nuclear Physics

Weiguang Jiang (蒋炜光)
UTK / ORNL
Machine Learning

- Logistic Regression
- Gaussian Processes
- Neural Network
- Support vector machine
- Random Forest
- Genetic Algorithm
Machine Learning
≈ Establish a function

• Image Recognition
  \( f(\quad) = "\text{Panda}" \)

• Playing Go
  \( f(\quad) = "3-4" \) (next move)

• Extrapolation
  \( f(\quad) = "\text{g.s.} : -27.63 \text{ MeV}" \)
Model:
- Complex function with lots of parameters (black box)

Image Recognition
\[ f(\text{Panda}) = \text{“Panda”} \]

\[ f_1(\text{Panda}) = \text{“Panda”} \]
\[ f_1(\text{Dog}) = \text{“Dog”} \]
\[ f_2(\text{Panda}) = \text{“Panda”} \]
\[ f_2(\text{Cat}) = \text{“Cat”} \]
Machine Learning
Supervised Learning

A set of function

\[
\begin{align*}
    f_1(x) &= \text{“Panda”} \\
    f_1(x) &= \text{“Dog”} \\
    f_2(x) &= \text{“Panda”} \\
    f_2(x) &= \text{“cat”} \\
    &\vdots
\end{align*}
\]

Goodness of the function \( f \)

Supervised Learning

Training Data

Function input: 
Function output: “bird” “Timon” “dog”
Machine Learning
Supervised Learning

A set of function $f_1, f_2, f_3, \ldots$

Goodness of the function $f$

Training Data

Training

Find the best function $f'$

Testing

“dog”

Use $f'$

“bird”  “Timon”  “dog”
Three Steps for Machine Learning

Step 1: Define a set of function

Step 2: Evaluate the function

Step 3: Choose the best function
Three Steps for Machine Learning

Step 1: Neural Network

Step 2: Evaluate the function

Step 3: Choose the best function
A sample of feed forward neural network (NN)
Feedforward Neural Network

Feedforward:
The value for every neuron only depend on the previous layer.

\[ z_1 = x_1 w_1 + \ldots + x_n w_n + \text{bias} \]

\[ z \rightarrow \sigma(z) \rightarrow x' \]

Activation function
Activation function

give non-linearity to the NN

\[ \sigma(z) = \frac{1}{1 + e^{-z}} \]

Sigmoid

\[ \sigma(z) \]

ReLU

\[ \sigma(z) = \max(0, z) \]

\[ a = z \]

\[ a = 0 \]
Neural Network Function

input layer (X): \( i \) neurons
one hidden layer: \( j \) neurons
output layer (Y): \( k \) neurons

\[
Z_j = \sum_i X_i W_{ij} + b_j
\]

\[
X'_j = \sigma (Z_j)
\]

\[
Y_k = \sum_j X'_j W'_{jk} + b'_k
\]
Tensor operation

Training data: \( p \) sample
input layer (\( X \)): \( i \) neurons
one hidden layer: \( j \) neurons
output layer (\( Y \)): \( k \) neurons

\[
Z_{(p,j)} = X_{(p,i)} \times W_{(i,j)} + b_{(p,j)}
\]

\[
\begin{pmatrix}
z_{11} & \cdots & z_{1j} \\
\vdots & \ddots & \vdots \\
z_{p1} & \cdots & z_{pj}
\end{pmatrix} =
\begin{pmatrix}
x_{11} & \cdots & x_{1i} \\
\vdots & \ddots & \vdots \\
x_{p1} & \cdots & x_{pi}
\end{pmatrix} \times
\begin{pmatrix}
w_{11} & \cdots & w_{1j} \\
\vdots & \ddots & \vdots \\
w_{i1} & \cdots & w_{ij}
\end{pmatrix} +
\begin{pmatrix}
b_{11} & \cdots & b_{1j} \\
\vdots & \ddots & \vdots \\
b_{p1} & \cdots & b_{pj}
\end{pmatrix}
\]

\[
Y_{(p,k)} = X'_{(p,j)} \times W'_{(j,k)} + b'_{(p,j)}
\]

\[
\begin{pmatrix}
y_{11} & \cdots & y_{1j} \\
\vdots & \ddots & \vdots \\
y_{p1} & \cdots & y_{pj}
\end{pmatrix} =
\begin{pmatrix}
\sigma(z)_{11} & \cdots & \sigma(z)_{1j} \\
\vdots & \ddots & \vdots \\
\sigma(z)_{p1} & \cdots & \sigma(z)_{pj}
\end{pmatrix} \times
\begin{pmatrix}
w'_{11} & \cdots & w'_{1j} \\
\vdots & \ddots & \vdots \\
w'_{i1} & \cdots & w'_{ij}
\end{pmatrix} +
\begin{pmatrix}
b'_{11} & \cdots & b'_{1j} \\
\vdots & \ddots & \vdots \\
b'_{p1} & \cdots & b'_{pj}
\end{pmatrix}
\]
Three Steps for Machine Learning

Step 1: Neural Network

Step 2: Evaluate the function

Step 3: Choose the best function
Evaluate a network

Introduce a **loss function** to describe the performance of the network (mse, cross entropy)

Loss:

\[ L = \sum_{r=1}^{p} l_p \]

Smaller the better

Image Recognition

Supervised:

\[ y_i = 0.2 \quad 0.8 \]
\[ \hat{y}_i = 0 \quad 1 \]

mse: \[ l_p = \frac{\sum_k (y_k - \hat{y}_k)^2}{k} \]
Three Steps for Machine Learning

Step 1: Neural Network

Step 2: Evaluate the function

Step 3: Choose the best function
“Learning” : find the best function

Ultimate goal:
Find the network parameters set that minimize the total loss $L$

Gradient Descent (even for AlphaGo)

- Compute $\frac{\partial L}{\partial w}$ with training data
- Update the parameters $w \leftarrow w - \eta \frac{\partial L}{\partial w}$
- Repeat until $\frac{\partial L}{\partial w}$ is small enough

This procedure is so-called machine learning.
Backpropagation

An efficient way to compute $\frac{\partial L}{\partial w}$

$$Z_j = \sum_i X_i W_{ij} + b_j$$

$$X'_j = \sigma(Z_j)$$

$$Y_k = \sum_j X'_j W'_{jk} + b'_k$$

mse: $l_p = \sum_k (y_k - \hat{y}_k)^2 / k$
Backpropagation

\[ Z_j = \sum_i X_i W_{ij} + b_j \]

\[ X'_j = \sigma(Z_j) \]

Sigmoid: \[ x' = \sigma(z) = \frac{1}{1+e^{-z}} \]

\[ Y_k = \sum_j X'_j W'_{jk} + b'_k \]

mse: \[ l_p = \sum_k (y_k - \hat{y}_k)^2 / k \]

\[ \frac{\partial l}{\partial w} = \frac{\partial l}{\partial z} \ast x \]

\[ \frac{\partial l}{\partial b} = \frac{\partial l}{\partial z} \]

\[ \frac{\partial l}{\partial z} = \frac{\partial l}{\partial x'} \ast \frac{1}{1+e^{-z}} \ast (1 - \frac{1}{1+e^{-z}}) \]

\[ \frac{\partial l}{\partial w'} = \frac{\partial l}{\partial y} \ast x' \]

\[ \frac{\partial l}{\partial b'} = \frac{\partial l}{\partial y} \]

\[ \frac{\partial l}{\partial y} = 2/k * (y - \hat{y}) \]
Optimizer

Gradient Descent:
walking in the desert, blindfold

Cannot see the whole picture

SGD
Momentum
AdaGrad
Adam
“Deep” learning

Deep just means more hidden layers

"Deep" is better

But not too deep

“Deep” VS “Wide”
“Deep” learning

Gradient vanishing/exploding problem

\( n' \) hidden layers
\( w_q \) is the weights for “q” \( q \) th hidden layer

\[
\frac{\partial l}{\partial x'} = \frac{\partial l}{\partial y} \cdot w
\]

\[
\frac{\partial l}{\partial w_q} = \frac{\partial l}{\partial y} \cdot w_n \cdot w_{n-1} \cdot w_{n-2} \cdots w_{q+1} \cdot x
\]
Overfitting

Training data and testing data can be different

Solution:
Get more training data
Create more training data
Dropout
L1/L2 regularization
Early Stopping
Friendly tool: Keras

- Python
- $ apt-get install python3-pip
- $ pip3 install keras
- $ pip3 install tensorflow
Neural network simulation & extrapolation

- NN application in nuclear physics
- $^4$He ground-state energy

Neural network simulation & extrapolation

- $^4$He radius
Neural network simulation & extrapolation

- The minimum g.s energy of each Nmax drops exponentially

Neural network for Coupled-cluster

\[ |\Psi_0\rangle = e^{\hat{T}} |\Phi_0\rangle \]

\[ \hat{T} = \sum_i T_i \]

\[ \hat{T}_{\text{CCSDT}} = \hat{T}_1 + \hat{T}_2 + \hat{T}_3 \]

CCSDT equations:

\[ \langle \Phi_i^a | e^{-T} H e^T | \Phi_0 \rangle = 0 \]
\[ \langle \Phi_{ij}^{ab} | e^{-T} H e^T | \Phi_0 \rangle = 0 \]
\[ \langle \Phi_{ijkl}^{abc} | e^{-T} H e^T | \Phi_0 \rangle = 0 \]

We want to truncate the 3p3h configurations

Introduce NN to select the more important configurations
Neural network for Coupled-cluster

- The input layer include all the quantum numbers of the 3p3h amplitudes \( (n \ l \ j \ j_{\text{tot}} \ t_z \ldots) \).

\[ ^{8}\text{He} \]

\[ ^{16}\text{O} \]
Neural network in nuclear matter

- Train with different combination of cD,cE.

![Graph showing energy per nucleon vs density](image-url)
**Neural network Uncertainty analysis**

- Even with the same input data and network structure, the NN will give different results in mutual independence trainings.

**Sources of uncertainty**

1. random initialization of neural network parameters
2. different divisions between training data and validation data
3. data shuffle (limit batch size)

**Solution**

1. ???
2. k-fold cross validation ...
3. increase batch size ...
Ensembles of Neural Networks

- $^4\text{He}$ radius
- The distribution of NN predict radius is Gaussian.
- Define the full width at half maximum value as the uncertainty for certain NN structure.
- The NN uncertainty reduce with more training data.
- The almost identical value of NN prediction radius indicates that the NN has a good generalization performance.
More complex case:

$^4$He g.s. energy, with two peak
$^4$He g.s. energy, with two peak

Nmax 4-16

Nmax 4-18

Nmax 4-20
We can separate the peaks with features (loss) provided by NN.