Machine Learning in Computational Chemistry: Classification Models, Tree Models, Introduction to Neural Networks

Jun Chen
Fujian Institute of Research on the Structure of Matter,
Chinese Academy of Sciences

July 14, 2025

Acknowledgements

Professor Dong Hui Zhang
(Dalian Inst. of Chem. Phys.)
who introduced me to the world of
neural network potentials in 2010s
(my fulfilling years as PhD candidates)

Experimental / theoretical collaborators:
Xueming Yang Group (DICP)
Xin Xu Group (Fudan Univ.)
Hua Guo Group (Univ. New Mexico)
constructed the first *ab-initio*-based NN potential
(fully converged to both fitting accuracy and dataset)

A global potential energy surface for the $H_2 + OH \leftrightarrow H_2O + H$ reaction using neural networks

Jun Chen,[1] Xin Xu,[1] Xin Xu,[2] and Dong H. Zhang[1,a)]
[1]*State Key Laboratory of Molecular Reaction Dynamics and Center for Theoretical Computational Chemistry, Dalian Institute of Chemical Physics, Chinese Academy of Sciences, Dalian 116023, People's Republic of China*
[2]*Department of Chemistry, Fudan University, Shanghai 200433, People's Republic of China*

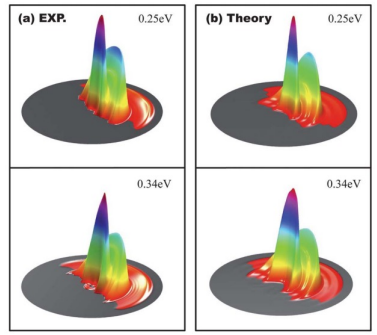
(Received 28 January 2013; accepted 28 March 2013; published online 17 April 2013)

J. Chem. Phys., 2013, 138(15): 154301.

The dynamics of the $D_2 + OH \rightarrow HOD + D$ reaction: A combined theoretical and experimental study

Shu Liu,† Chunlei Xiao,† Tao Wang, Jun Chen, Tiangang Yang, Xin Xu, Dong H. Zhang* and Xueming Yang*

Received 10th February 2012, Accepted 19th March 2012
DOI: 10.1039/c2fd20018j

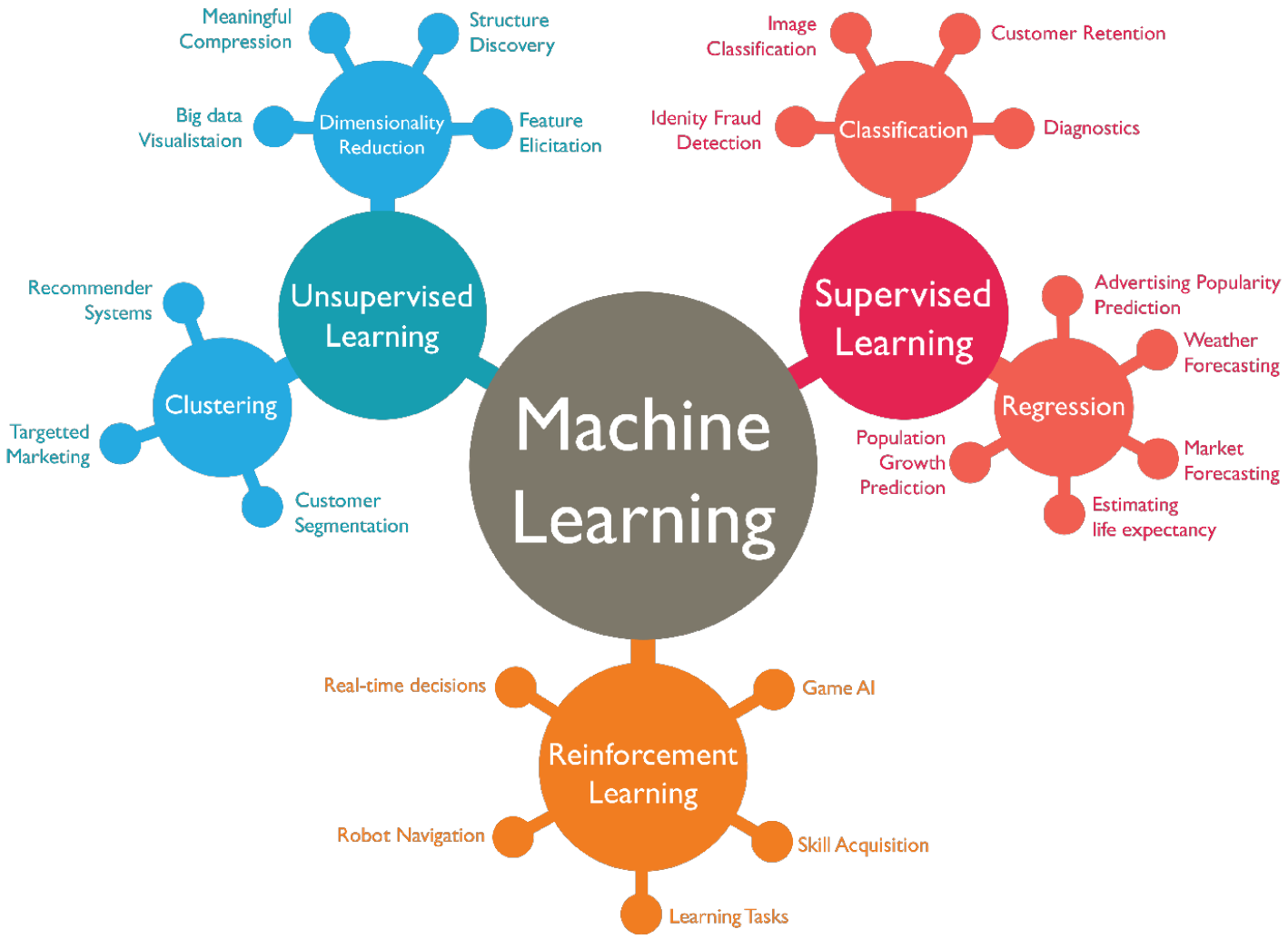


Faraday Discussions, 2012, 157: 101-111.

Our goals today:

- **Familiar with classification and regression**
- **Familiar with SVM/RF/xGB/NN methods**
- **Become an expert in GP/NN PES construction!** (after the practical session)

Part A - algorithms



➢ Classification Models

Origin of classification models?

Classification models start with statistics(discriminant analysis) and pattern recognition(early AI) Key milestones: Logistic Regression and Naive Bayes.

Brief History:

- 1950s: Founding of statistics: Fisher's Linear Discriminant Analysis.
- 1980s-1990s:Pattern recognition emerges: Decision trees, KNN, Support Vector Machines(SVM).
- 2000s: Integration of Materials Science and AI: Materials Genome Initiative (High-throughput Data-Driven).
- 2010s: Research on reshaping the classification model of deep learning.

What is Classification Models?

- **Definition:** a supervised learning method used to classify data points into categories.
- **Output:** Categorical (e.g., 'active'/'inactive', 'metal'/'insulator', 'stable'/'unstable')
- **Contrast with Regression:**Regression predicts a continuous value (e.g., binding affinity, band gap energy, potential energy).

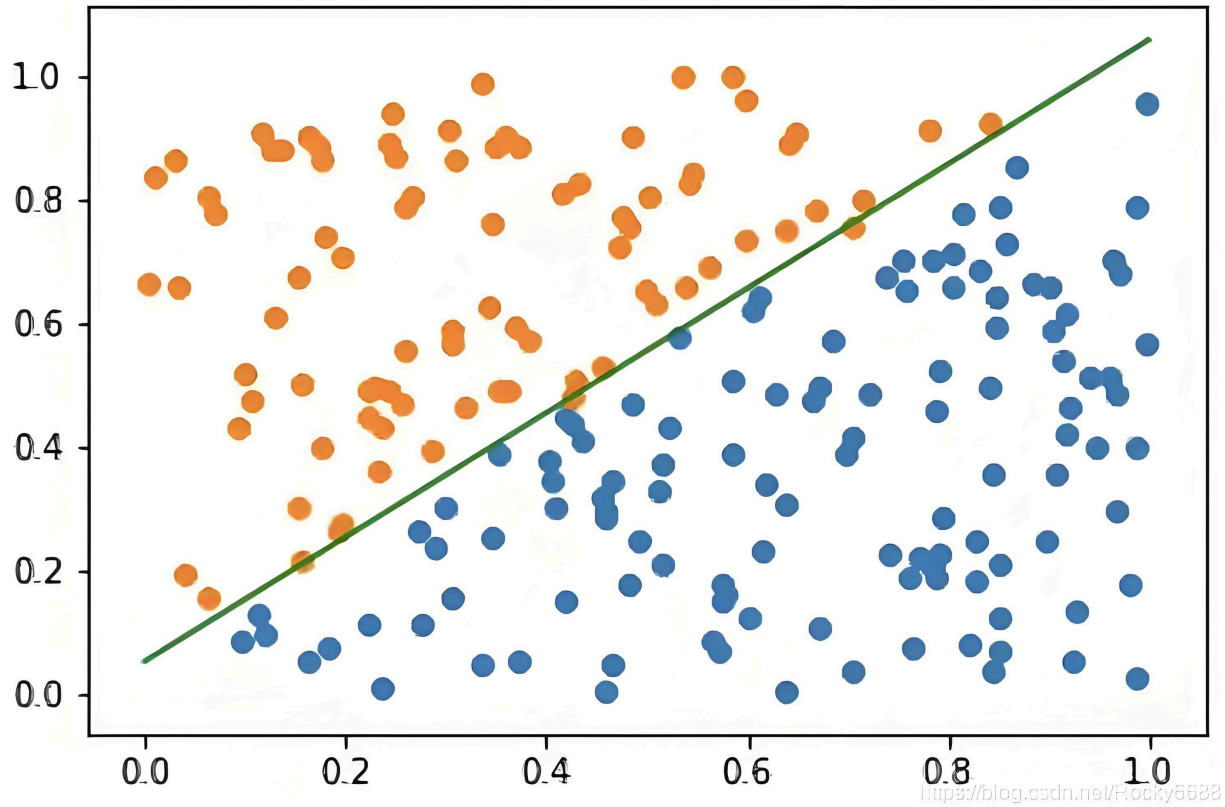


Figure:Illustration of Binary Classification (Imagine separating red and blue points with a line)

- **Core idea:** By optimizing algorithms (such as minimizing the loss function), the model parameters are adjusted so that the decision boundary can best distinguish between different types of sample points.

Common Algorithms: Logistic Regression (逻辑回归)

- Logistic regression is suitable for predicting binary categorical variables.
- It maps the output of a linear function to a probability between 0 and 1 through the sigmoid function.

Sigmoid Function:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

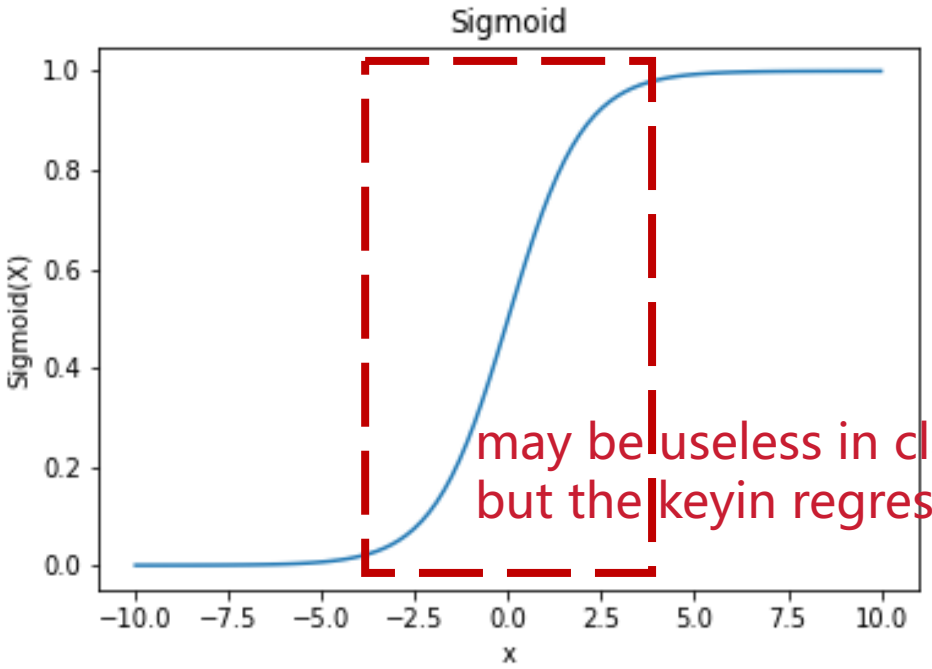


Figure:Sigmoid Function Plot
(S-shaped curve from 0 to 1)

Prediction Equation

For binary classification (y∈{0,1}):

$$P(y = 1/x) = \sigma(w^T x + b) = \frac{1}{1 + e^{-(w^T + b)}}$$

w: weight vector

b: bias term

x: material characteristic vector

- **Advantages:**
  The computational cost is low, and it is easy to understand and implement.
- **Disadvantage:**
  It is prone to underfitting and the classification accuracy may not be high.
- **Applicable data types:**
  Numeric Data and Nominal Data

Common Algorithms: Support Vector Machines (SVM, 支持向量机)

- Finds the **optimal hyperplane** that best separates data points into different classes.
- Key: Maximize the **margin** between the decision boundary and the nearest training data points (support vectors).
- Handles both linearly separable and non-linearly separable data (using **kernel trick**).

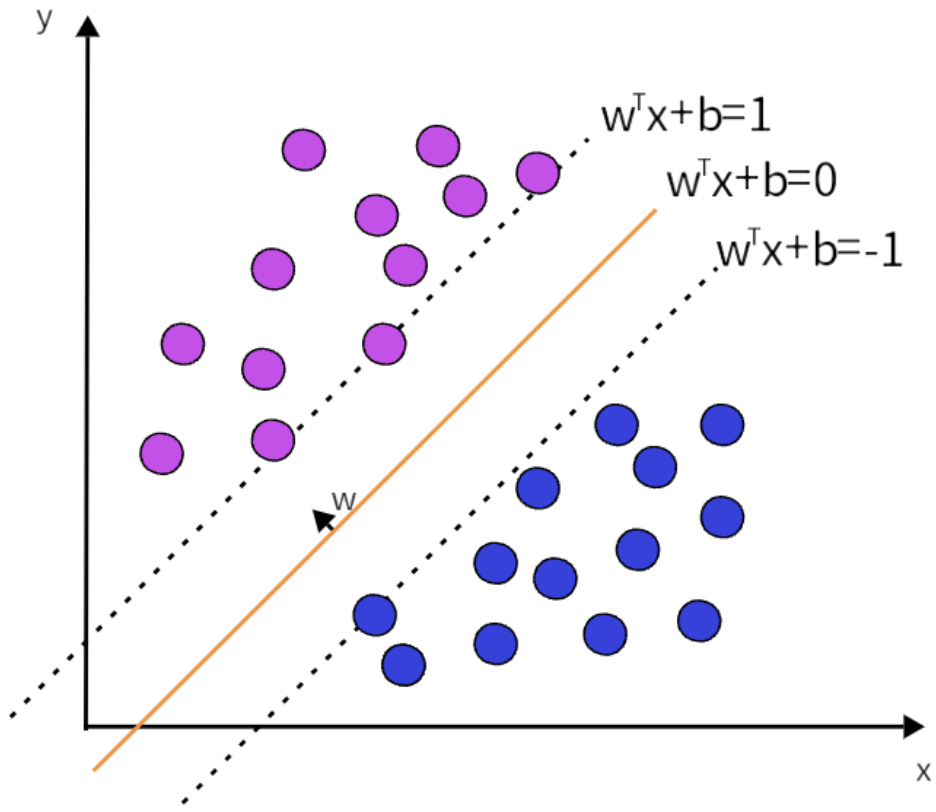


Figure:SVM: Maximizing the Margin (Imagine two classes of points, a separating line, and parallel lines defining the margin)

Key Concepts

- **Hyperplane:** $w^T x + b = 0$
- **Support Vectors:** Data points closest to the hyperplane.
- **Kernel Trick:** Implicitly maps data into higher-dimensional space where it might be linearly separable (e.g., Radial Basis Function(RBF).)

- **Advantages:**
  Low generalization error rate, low computational cost, and easily interpretable results.
- **Disadvantages:**
  Sensitive to parameter adjustment and choice of kernel function. The original classifier, without modification, is only applicable for handling binary problems.
- **Applicable data types:**
  Numeric Data and Nominal Data

Evaluation Metrics for Classification

Key Performance Indicators

Indicators	Formulas	Scientific significance	Ideal value
Accuracy	(TP+TN)/(TP+FP+FN+TN)	Overall prediction accuracy rate	Nearly 1
Precision	TP/(TP+FP)	The proportion of true stable materials in the predicted stable materials	Nearly 1
Recall	TP/(TP+FN)	The proportion of truly stable materials that are correctly identified	Nearly 1

TP (True Positive) TN (True Negative)

FP (False Positive) FN (False Negative)

Evaluation Metrics for Classification

The confusion matrix is a tool used to evaluate the performance of classification models. The confusion matrix compares the classification results with the actual labels to illustrate the prediction performance of the model.

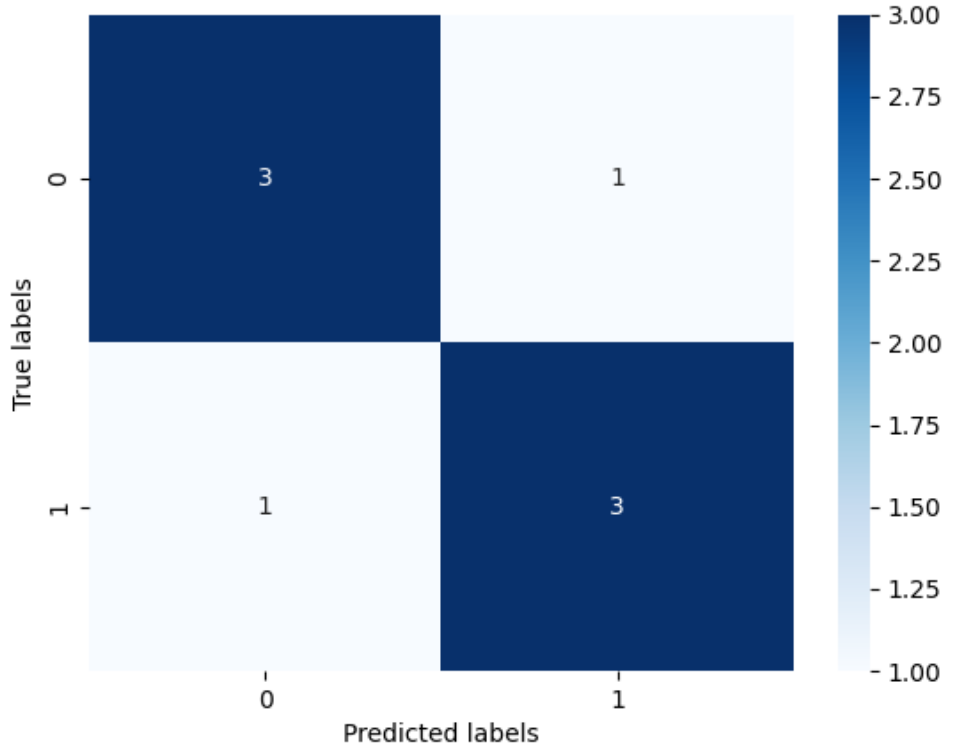


Figure:Confusion Matrix

Application fields

- High-Throughput Virtual Screening (HTVS)
  - e.g. Lithium battery electrolyte
  - Input: Ion conductivity + Electrochemical window
  - Output: Safe/Unstable Classification → Experimental Verification Efficiency Increased by 5 Times
- Structural Classification
  - e.g. Molecular dynamics trajectory analysis
  - Input: Local atomic coordinates
  - Output: Identification of amorphous/crystalline regions (SVM accuracy > 92%)
- Performance Prediction
  - e.g. Classification of catalyst activity levels
  - Input: Surface adsorption energy + d-band center
  - Output: High/Middle/Low Activity → Guide Experimental Synthesis
- Synthetic Path Optimization
  - e.g. Prediction of MOF Synthesis Conditions
  - Input: Solvent polarity + Reaction temperature
  - Output: Probability of success/failure → Reduction of trial-and-error costs

➢ Tree Models

Origin of tree models?

Definition: The tree model is a supervised learning algorithm based on a tree structure. It predicts by recursively dividing the feature space into mutually exclusive regions.

- 1960s: The basic idea of decision tree is proposed.
- 1975: ID3: The use of the information gain criterion to select the splitting attribute marked the official birth of the decision tree classification algorithm.
- 1980s-1990s: C4.5, CART and the seeds of the idea of ensemble learning began to germinate.
- 1995: Random Forest: Improve the performance of tree ensemble, significantly enhancing the performance of classification and regression tasks.
- 2007: Gradient Boosting Machines (GBM) : The Boosting tree model framework based on gradient descent to optimize the loss function has significantly improved the accuracy.
- 2010s: The rise of XGBoost.
- After 2016: LightGBM and CatBoost were released, improving training speed and accuracy. LightGBM innovatively adopted an efficient splitting algorithm based on histograms, while CatBoost addressed the bias of categorical features.

Decision Trees- The Basics

- A flowchart-like structure where each internal node represents a ”test” on an attribute (feature).
- Each branch represents an outcome of the test.
- Each leaf node represents a class label (decision).
- Works by recursively splitting the data based on features to maximize homogeneity (purity) within each child node.

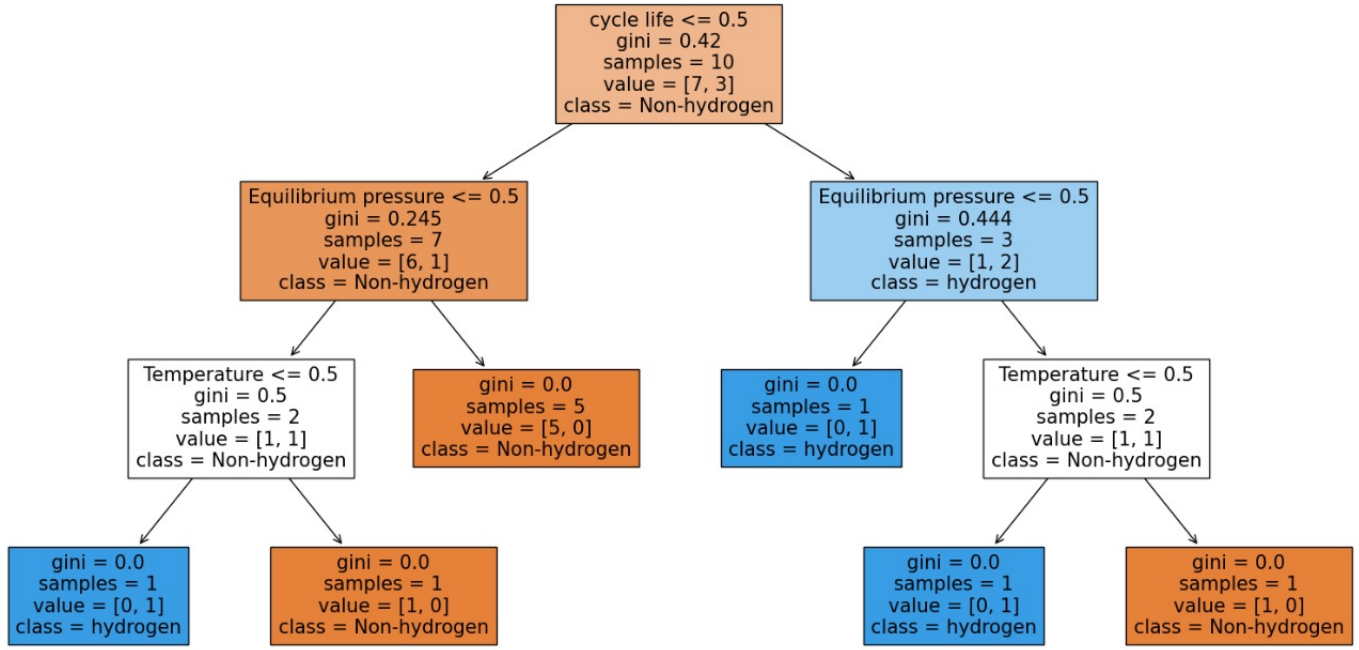


Figure:Decision Tree for Material Classification

Pros & Cons

- **Pros:** Highly interpretable, easy to understand.
- **Cons:** Prone to overfitting, sensitive

Building a Decision Tree: Gini Impurity

- At each node, we select the feature and split point that best divides the data.
- **Gini Impurity** is a common metric to quantify the "purity" or "mixed-up-ness" of a node.
- A node is pure if all its samples belong to the same class (Gini=0).

Gini Impurity Formula

For a node p with C classes, where p_k is the proportion of samples belonging to class k:

$$G(p) = 1 - \sum_{k=1}^{c} (p_k)^2$$

Example Calculation

Consider a node with 10 samples: 7 are Class A, 3 are Class B.

$p_A = 7/10 = 0.7$, $p_B = 3/10 = 0.3$

$G = 1 - ((0.7)^2+(0.3)^2) = 1 - (0.49+0.09) = 1 - 0.58 =$ **0.42**

Building a Decision Tree: Information Gain (Gini Gain)

- We want to find the split that maximizes the reduction in impurity.
  This reduction is called **Information Gain** or **Gini Gain**.
- The algorithm iterates through all possible features and split points to find the best one.

Information Gain Formula for a Split

For a parent node S split into m child nodes S_v based on attribute A:

$$Gain(S, A) = G(s) - \sum_{v \in Values(A)} \frac{|S_v|}{|s|} G(S_v)$$

Where $|S_v|$ is the number of samples in child node S_v, and $|S|$ is the number of samples in the parent node S.

Building a Decision Tree: Information Gain (Gini Gain)

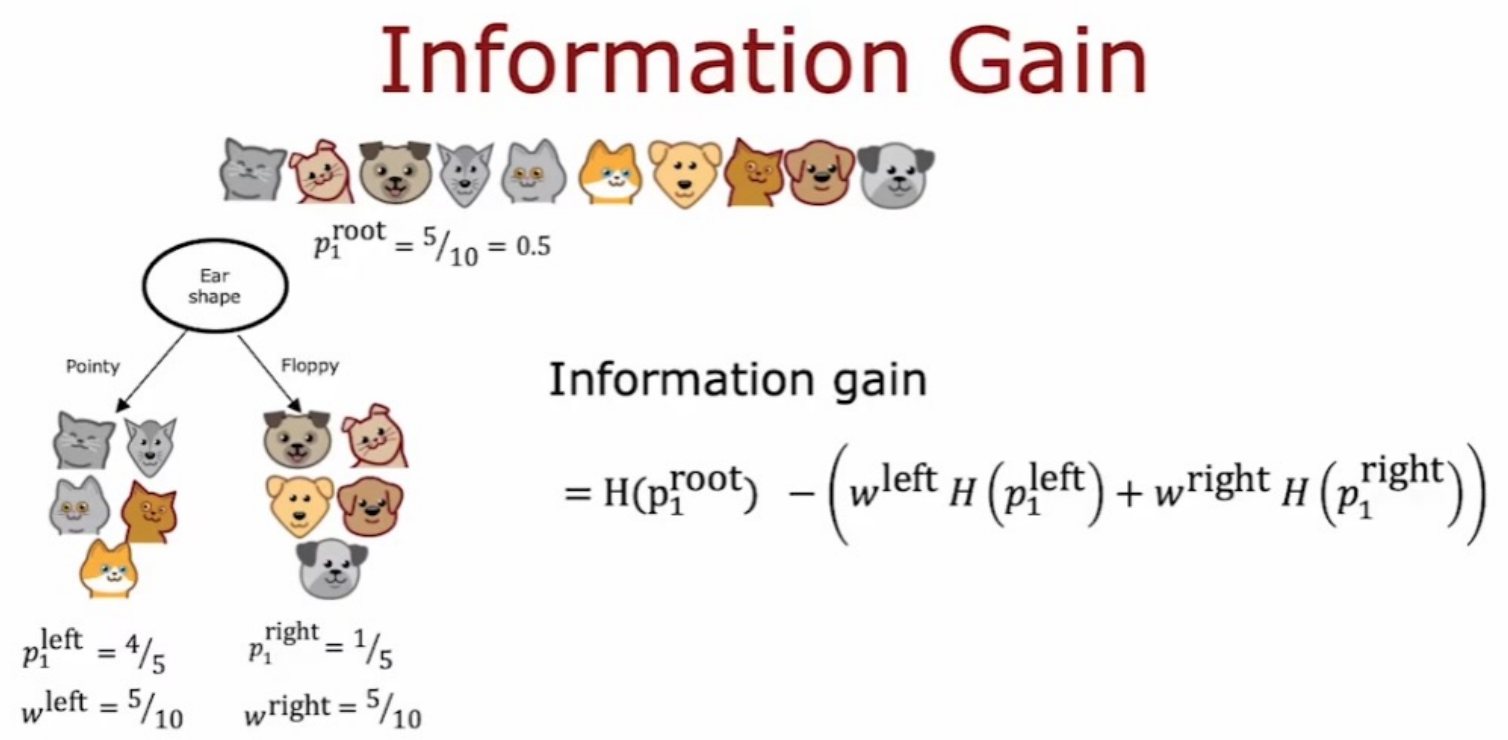


Figure: Information Gain

Ensemble Methods: Beyond Single Trees

- Single decision trees are often prone to overfitting and can be unstable.
- Ensemble methods combine the predictions of multiple individual models (often ”weak learners”) to produce a more robust and accurate model.
- Two main types:
    - **Bagging** (Bootstrap Aggregating): Build multiple models independently and average their predictions (e.g., Random Forests).
    - **Boosting:** Build models sequentially, where each new model corrects the errors of the previous ones (e.g., Gradient Boosting).

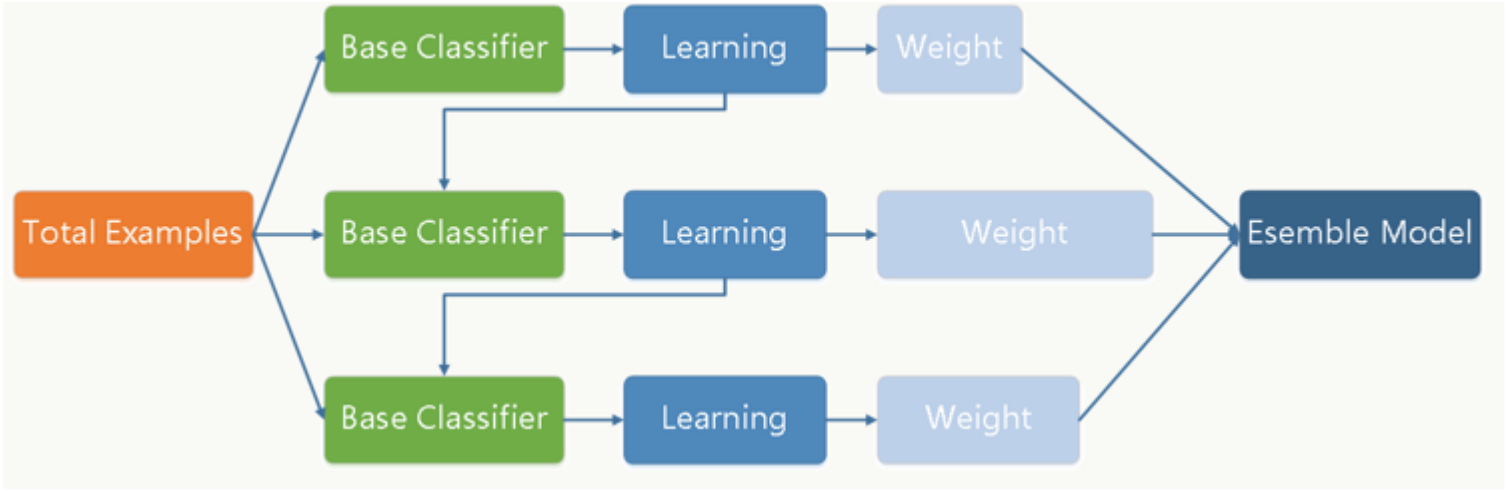


Figure:Ensemble Learning Concept (Imagine multiple decision trees contributing to a final decision)

Random Forests

An ensemble learning method for both classification and regression. Builds multiple decision trees and merges their predictions to get a more accurate and stable prediction.

Two key randomization techniques:

1. Bagging (Bootstrap Aggregating): Each tree is trained on a different random subset (with replacement) of the training data.
2. Feature Randomness: At each split in a tree, only a random subset of features is considered.

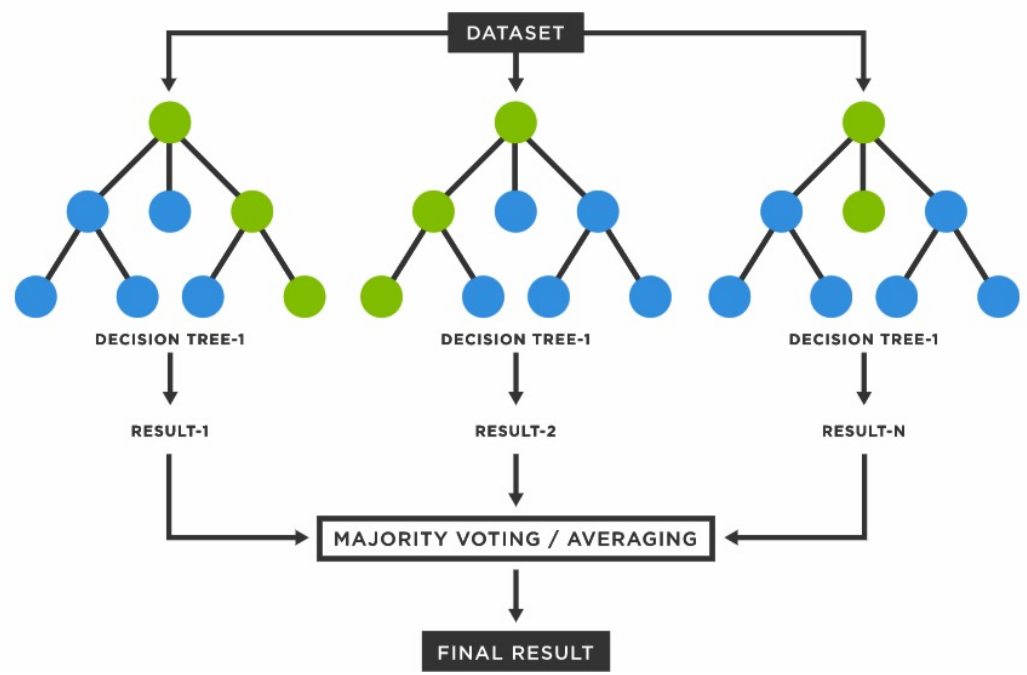
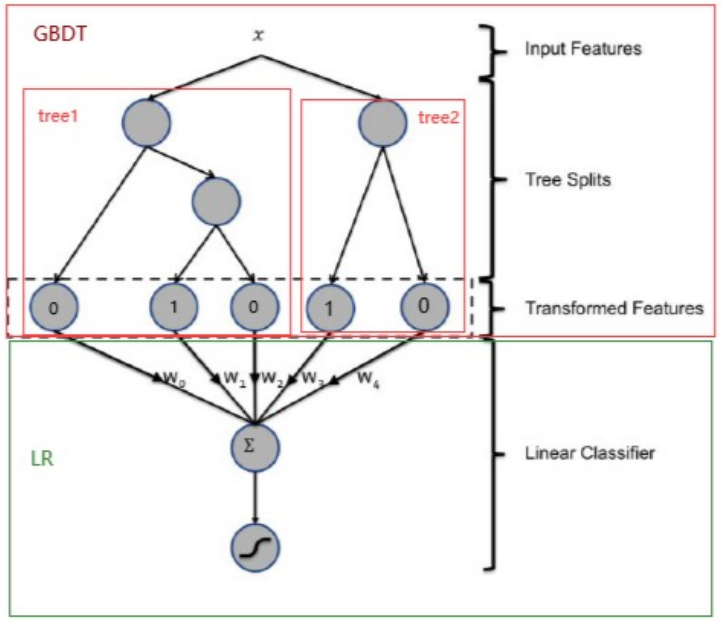


Figure:Random Forests

Gradient Boosting (e.g., XGBoost, LightGBM)

An ensemble method where new models are added to sequentially correct the errors made by previous models.
Builds trees one at a time, where each new tree tries to minimize the loss function of the previous model by learning from the “residuals” (errors).
How it works (intuition):

1. Start with an initial simple model (e.g., a single leaf predicting the average).
2. Calculate the “residuals” (errors) of this model.
3. Train a new weak learner (decision tree) to predict these residuals.
4. Add the predictions of this new tree to the previous model’s predictions, scaled by a learning rate.
5. Repeat steps 2-4 for many iterations



XGBoost vs LightGBM vs Random Forest Performance Comparison

Characteristic	XGBoost	LightGBM	Random Forest
Training Speed	Medium	**Fast**	Slow
Prediction Accuracy	**High**	Medium-High	Medium
Memory Usage	Medium	**Low**	High
Overfitting Risk	Medium-Low	Medium	**Low**
Interpretability	Medium	Medium	**High**
GPU Support	**Excellent**	**Excellent**	Limited
High-Dim Data Handling	Excellent	**Excellent**	Good

Application fields

1. Perovskite solar cell screening

Method: XGBoost classifier.

Input: Tolerance factor + Octahedral factor Output: Stable / Unstable.

Result: 142 viable experimental materials were selected from 120,000 candidates.

2. Classification of semiconductor point defects

Input: Defect formation energy + Charge transfer quantity.

Output: Reorganization center / Non-radiation center.

Key finding: The SHAP values reveal that the S impurity is the main recombination center.

3. Prediction of MOF synthesis success rate

Characteristics: Solvent polarity + Reaction temperature + Ligand length.

Model: LightGBM classifier (AUC = 0.91).

Application: Reduces failed experiments by 63%.

4. Classification of Molecular Dynamics Trajectories

(1) MD simulation

(2) Extract local atomic environment

(3) Random forest classification

(4) Identify BCC/HCP/Amorphous regions

The accuracy rate is 98.7%, which is 17 times faster than SVM.

➢ Neural Networks

Biological Inspiration & History

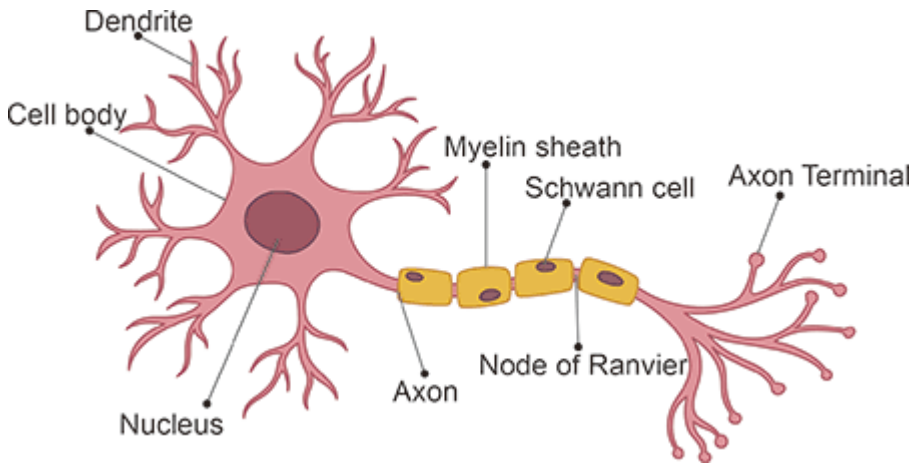


Figure: A Biological Neuron

Biological Neurons:

- Receive signals from dendrites.
- Process signals in the cell body.
- Transmit signals via axon to other neurons.
- Synapses adjust strength of connections.

Brief History:

- **1943:** McCulloch and Pitts propose the first computational model of a neuron.
- **1957:** Frank Rosenblatt creates the **Perceptron**, the first functional artificial neural network.
- **1969:** Minsky and Papert's "Perceptrons" highlights limitations (e.g., XOR problem), leading to "AI Winter".
- **1980s:** Backpropagation algorithm rediscovered/popularized, enabling training of multi-layer networks

The Artificial Neuron (Perceptron)

- The fundamental building block of a Neural Network.
- Mimics the biological neuron's function:
  - Receives multiple numerical **inputs** (x_i).
  - Each input is multiplied by a corresponding **weight** (w_i).
  - A **bias** (b) term is added.
  - The sum is passed through an **activation function** (f).
  - Produces a single **output**.

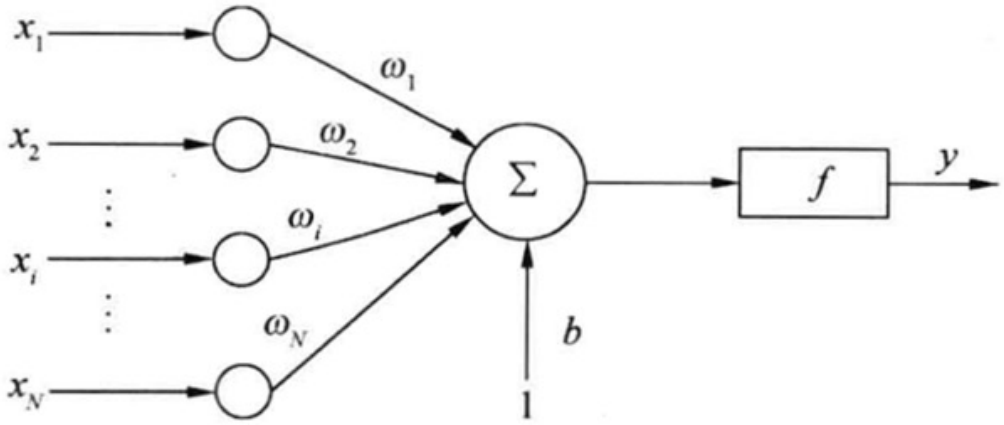


Figure: Artificial Neuron (Perceptron)

Mathematical Formulation

Weighted Sum: $z = \sum_{i=1}^{n} w_i x_i + b = w^T x + b$

Output (Activation): $a = f(z)$

Activation Functions

- Introduce non-linearity into the network, allowing it to learn complex patterns.
- Without them, a neural network would just be a series of linear operations, equivalent to a single linear model.

Sigmoid (Logistic)

- Formula: $\sigma(z) = \frac{1}{1+e^{-z}}$
- Range: (0, 1)
- Pros: Historically important, provides probabilities.
- Cons: Vanishing gradient problem for very large or very small z.

Rectified Linear Unit (ReLU)

- Formula: f (z) = max(0, z)
- Range: [0, ∞)
- Pros: Computationally efficient, helps mitigate vanishing gradient.
- Cons: ”Dying ReLU” problem (neurons can become inactive).

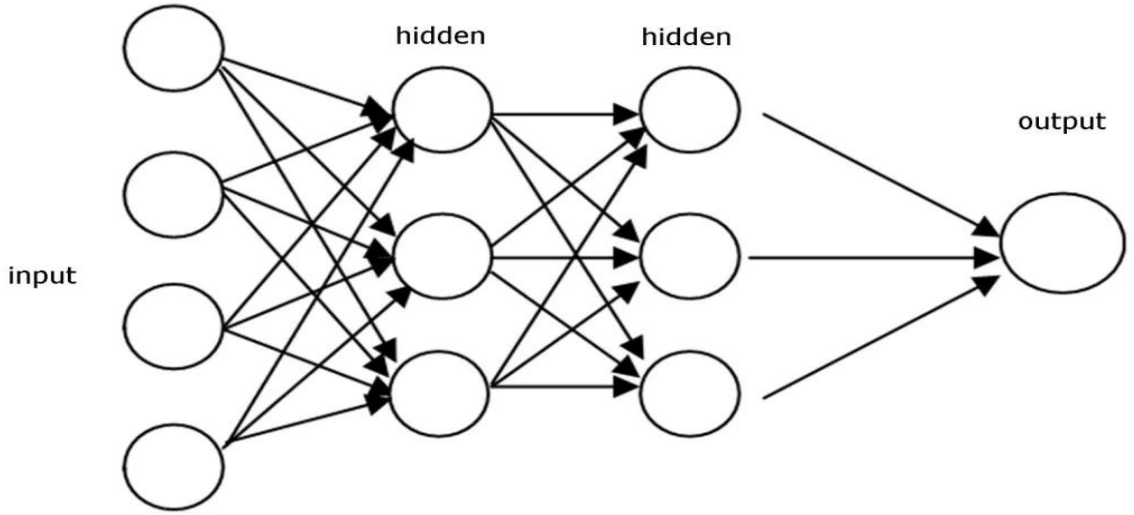
- **Others:** Tanh ($\tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$), Leaky ReLU, ELU, Swish.

Common activation functions and their derivatives

Activation Functions	Functions	derivatives
Logistic Functions	$f(x)=\frac{1}{1+\exp(-x)}$	$f'(x)=f(x)(1-f(x))$
Tanh Functions	$f(x)=\frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$	$f'(x)=1-f(x)^2$
ReLU Functions	$f(x)=\max(0,x)$	$f'(x)=I(x>0)$
ELU Functions	$f(x)=\max(0,x)+\min(0,\gamma(\exp(x)-1))$	$f'(x)=I(x>0)+I(\leq 0)\cdot\gamma\exp(x)$
SoftPlus Functions	$f(x)=\log(1+\exp(x))$	$f'(x)=\frac{1}{1+\exp(-x)}$

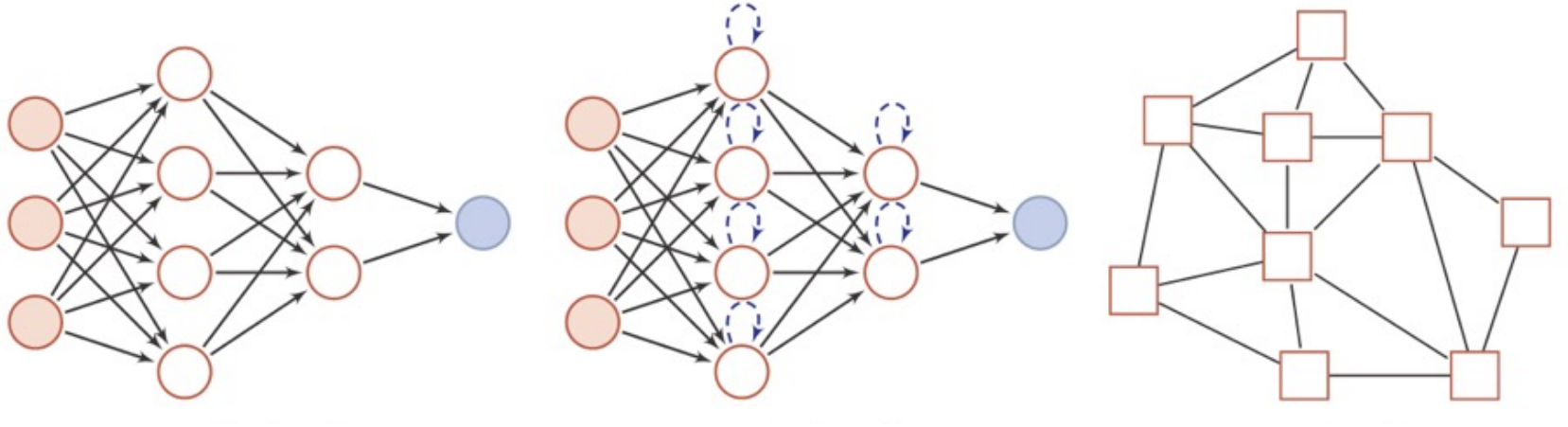
Feedforward Neural Networks (Multilayer Perceptrons - MLP)

- Neurons are organized into layers.
- **Input Layer:** Receives raw data. No computation performed.
- **Hidden Layers:** One or more layers between input and output. Perform computations, learning increasingly complex representations.
- **Output Layer:** Produces the final prediction (e.g., class probabilities for classification, a single value for regression).
- **Feedforward:** Information flows in one direction, from input to output, without loops.



Memory networks and networks with graphs

- **Memory networks,** also known as feedback networks, are networks in which neurons not only receive delicate bits of information from other neurons, but also from their own histories.
- **Graph networks** are neural networks defined on graph-structured data, where each node in the graph consists of a neuron or a set of neurons, the connections between nodes can be directed or undirected, and each node can receive information from neighboring nodes or from itself. Graph neural networks are a generalization of feedforward and memory neural networks.



Neural Network Implementation Core Components

subassemblies	Calculate network output	Realization points
forward propagation	Calculate network output	Matrix multiplication and activation functions
loss function	Quantifying prediction error	Numerical stability needs to be considered
backward propagation	Calculate the parameter gradient	Recursive applications of the chain rule
Parameter update	Optimization weight	Optimizer Selection

Universal Approximation Theroem

- Feedforward neural networks have a strong fitting ability, and common continuous nonlinear functions can be approximated by feedforward neural networks

$$F(x)=\sum_{m=1}^{M}\nu_m\phi(\omega_m^T x+b_m)$$

approximation

$$\left|F(x)-f(x)\right|<\varepsilon,\forall x\in J_D$$

Training a Neural Network: Backpropagation (Intuition)

- The goal of training is to find the optimal weights and biases that minimize the difference between predicted and actual outputs.
- This difference is quantified by a **Loss Function** (or Cost Function).
    - **Regression:** Mean Squared Error (MSE):

$$\frac{1}{N}\sum(y_i - \hat{y}_i)^2$$

    - **Classification:** Cross-Entropy Loss (for probabilities).
- **Gradient Descent:** An iterative optimization algorithm that adjustsweights and biases in the direction that most steeply reduces the loss function.

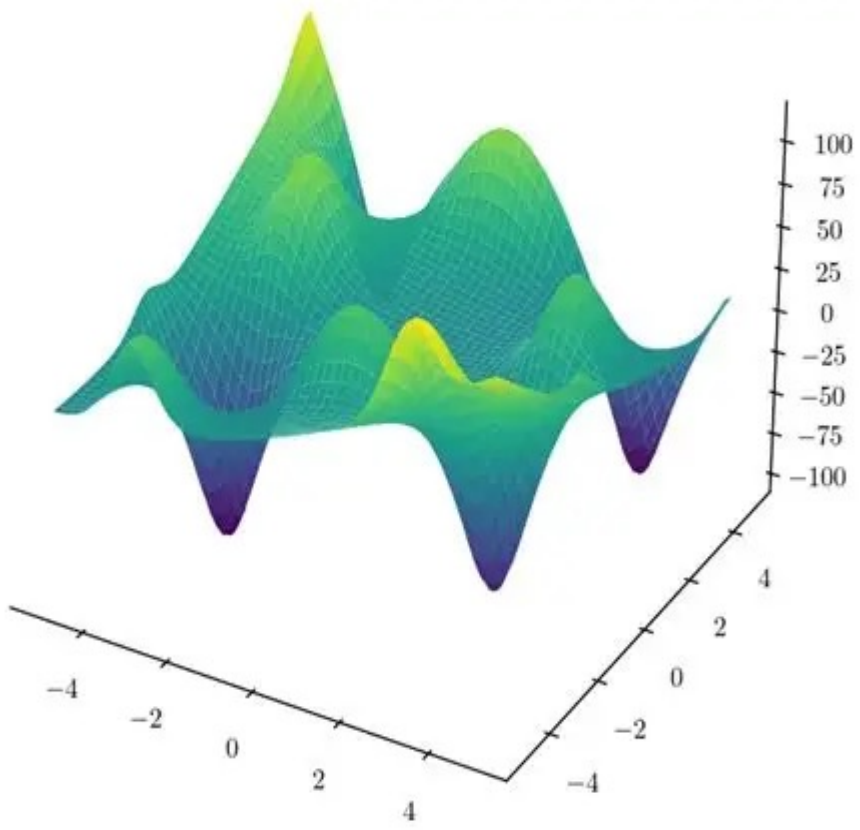


Figure: Gradient Descent: Finding the Minimum of a Loss Function (Imagine a ball rolling down a hill to the lowest point)

Training a Neural Network: Backward propagation (Algorithm)

- Backward propagation is the core algorithm for training ANNs. It uses the chain rule of calculus to efficiently compute the gradients of the loss function with respect to every weight and bias in the network.

- Two Phases:
  - Feed forward: Input data flows through the network, layer by layer, to compute the output and the loss.
  - Backward propagation: The error (difference between predicted and actual) is propagated backward through the network. Gradients are calculated for each weight and bias, indicating how much they contributed to the error.

NN = ANN = FFNN = BPNN

sometimes, BPNN = Behler-Parrinello NN

Jörg Behler and Michele Parrinello, Phys. Rev. Lett. 98, 146401 (2007)

Training a Neural Network: Backpropagation (Algorithm)

Weight Update: Weights and biases are then adjusted using the calculated gradients and a learning rate (α):

$$W_{new} = W_{old} - \alpha \frac{\partial L}{\partial W}$$

This process is repeated over many ”epochs” (passes through the entire dataset)

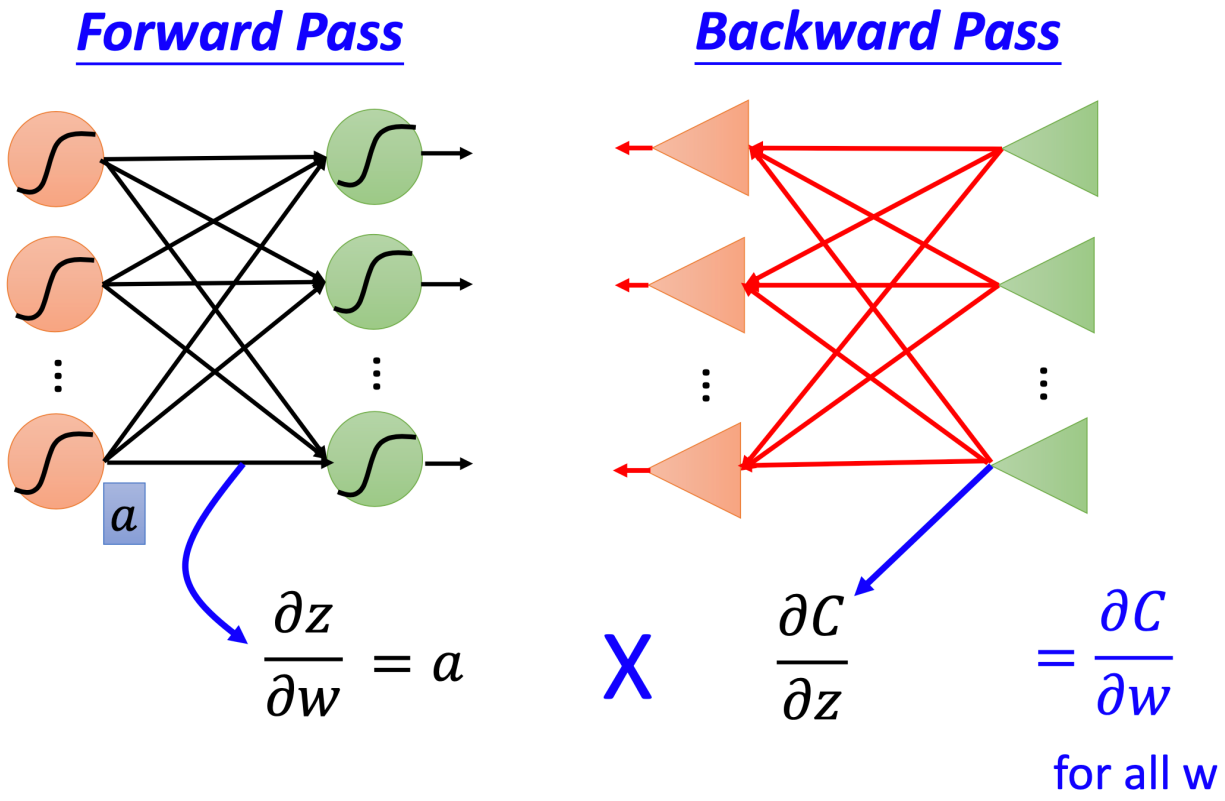


Figure: Forward and Backward Pass in Backpropagation

Key Hyperparameters in Neural Networks

- **Learning Rate (α)**: Determines the step size at each iteration while moving toward a minimum of the loss function. (Too high:overshoot; Too low: slow convergence).
- **Number of Hidden Layers:** Depth of the network. More layers enable learning more complex hierarchical features.
- **Number of Neurons per Layer (Width):** Capacity of the layer to learn representations.
- **Activation Functions:** (Discussed previously: ReLU, Sigmoid, Tanh, etc.). Chosen based on task and layer.
- **Epochs:** Number of times the entire training dataset is passed
- through the neural network.
- **Batch Size:** Number of training examples utilized in one iteration. (Larger batches: stable gradients, less frequent updates; Smaller batches: noisy gradients, more frequent updates).
- **Optimizers:** Algorithms (e.g., Adam, SGD, RMSprop) that control how weights and biases are adjusted based on gradients.

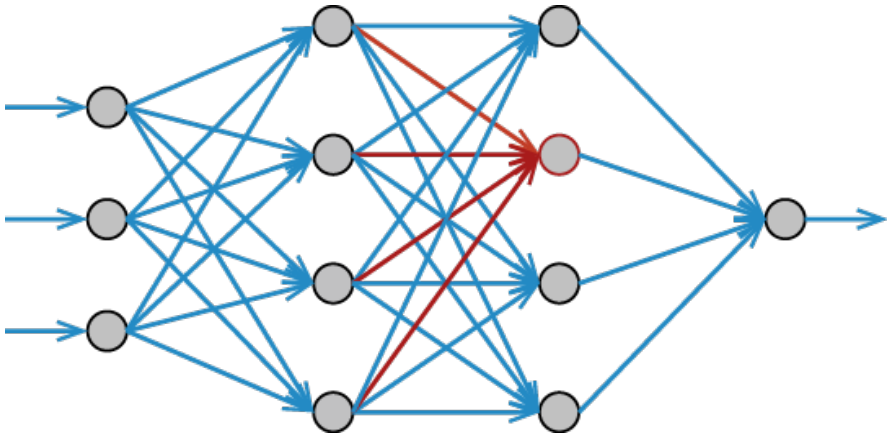
Training a Neural Network: Backpropagation (Cont.) - Error FunctionDerivation

A Simplified Example: Single-Layer Network

Let's walk through the weight update process using backpropagation with a simple single-layer neural network.

Network Structure:

- Single input x
- Single neuron with weight w and bias b
- Activation function σ(z) (e.g., Sigmoid)
- Output $\hat{y} = \sigma(wx + b)$
- Target output y



Error Function (Loss Function):

We'll use a simple Mean Squared Error (MSE) as our loss function L:

Applications of Neural Networks

- Molecular Property Prediction and Virtual Screening

  Neural networks can rapidly predict physicochemical properties (e.g., solubility, logP value) and biological activities (e.g., IC50 value) of tens of thousands of compounds by directly learning the relationship between molecular structure (e.g., SMILES string, molecular diagram, or 3D conformation) and properties.

- Chemical reaction prediction and synthetic route design

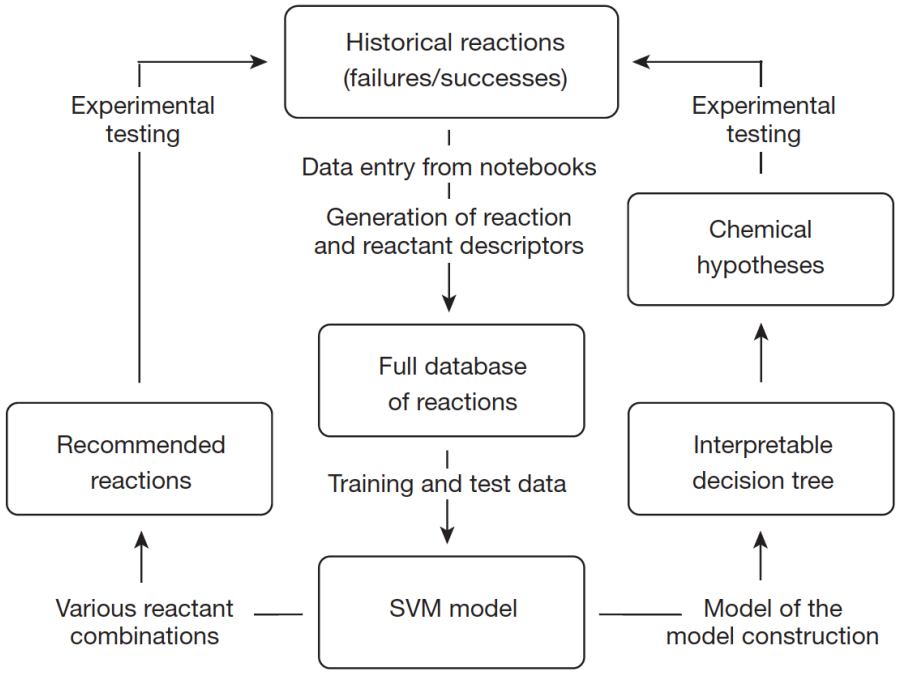
  Transformer-based neural networks (e.g., Molecular Transformer) can predict the product distribution and optimal reaction conditions of chemical reactions with an accuracy of over 90%.

- Molecular Dynamics Simulation Acceleration

  Conventional molecular dynamics (MD) simulations are limited by computational cost, making it difficult to study millisecond-scale processes. Neural Network Potential (NNP) can realize nanosecond simulations with DFT-level accuracy by fitting quantum mechanical calculations.

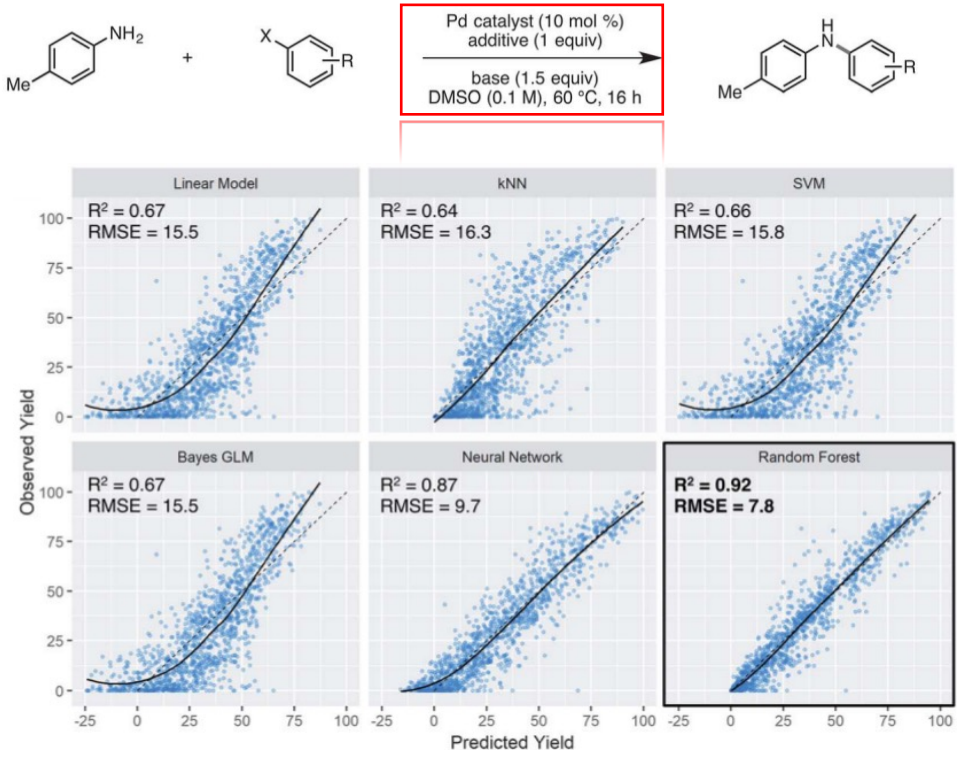
Literatures for reading

Importance of failed experiments



Raccuglia et al. *Nature* **2016**, *533*, 73.

Yield prediction of Buchwald-Hartwig reaction



Ahneman et al., *Science* **2018**, *360*, 186

Part B - NN potentials (NNP, MLP, MLFF)

$$\hat{H}(r_e, R_N) = \hat{T}_e + \hat{T}_N + V(r_e, R_N)$$

The whole Schrödinger equation of chemistry

Fix configuration

$$\left[\hat{T}_e + \mathrm{V}\right]\Phi_n(R_N \mid r_e) = E_n(R_N)\Phi_n$$

BO appr. ab initio calculation

$$E_0(R_i)$$

Construct PES

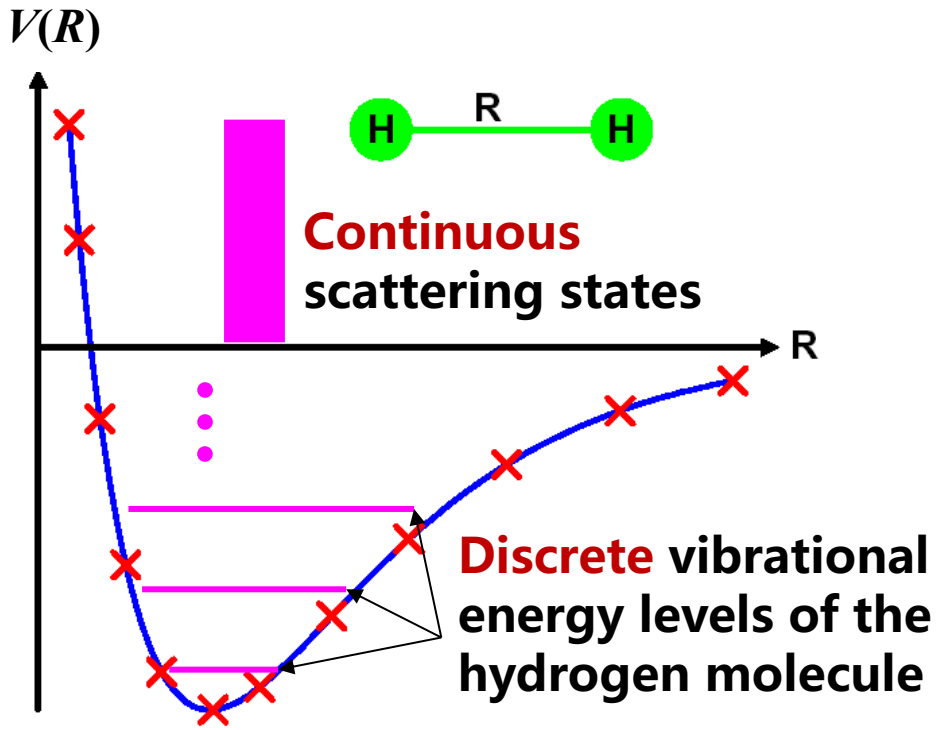
PES: $V(R)$

motion for nucleus

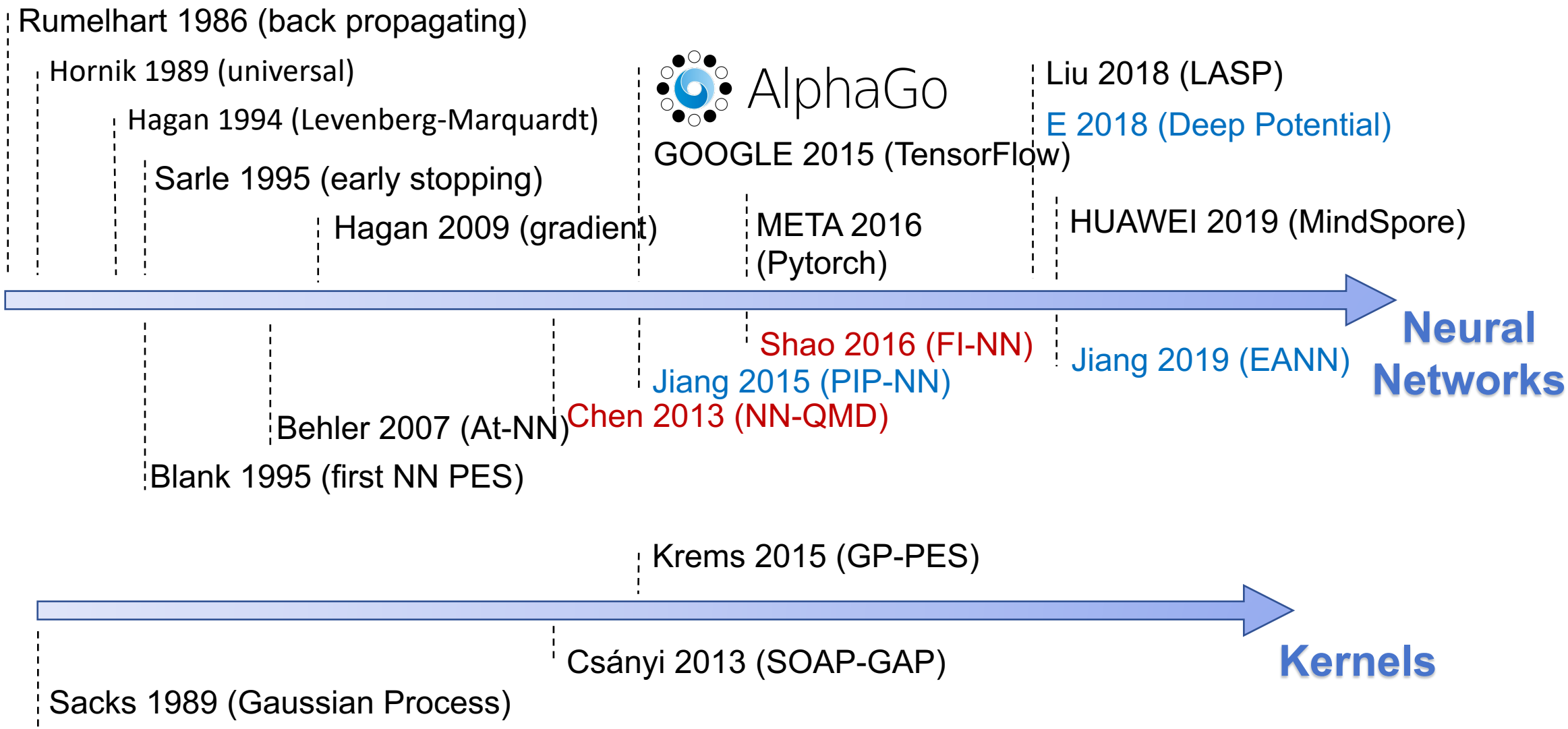
$$\left[\hat{T}_N + V(R)\right]\Psi(R) = E\Psi(R)$$

theoretical study of molecular dynamics :

- **electrons : construction of PES**
- **nucleus : equation of motion**



The rising of Machine Learning Potentials



A brief history of NN-PES

Previous PES methods

- spline — low dimensions
- Shepard inter. — slow evaluation
- Empirical Function — low accuracy
- polynomials — medium accuracy

• NN-based fittings

History of NN PES

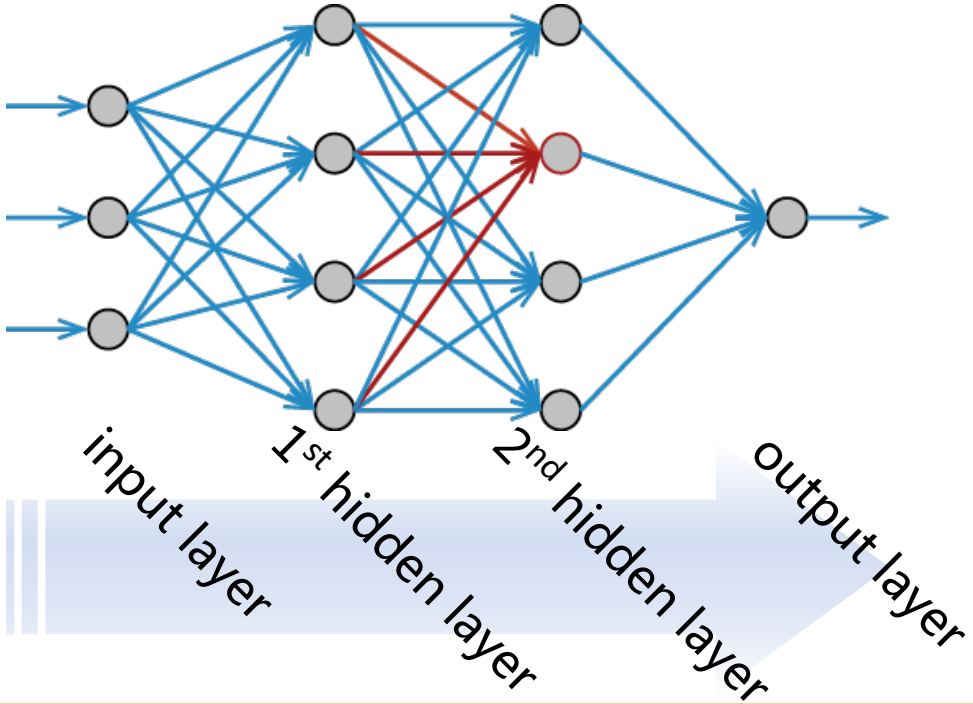
1994 LM algorithm：by Hagan
1995 avoid over fitting：by Sarle
2009 study gradients：by Hagan

1995 H2/Si(100), the first NN-PES
2006 H_2CO, HOOH <1meV

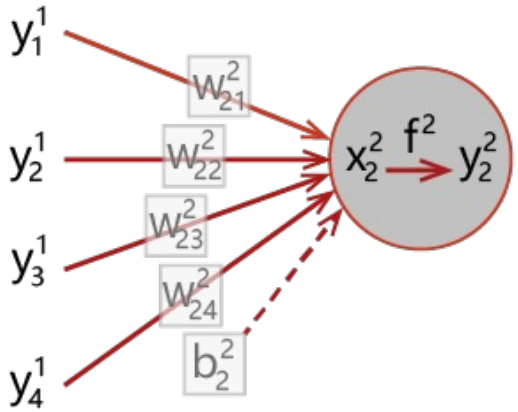
2002 NN ensembles
2006 Nested
2007 the atomic-NN

Functional type of neural networks

- the feedforward NN with two hidden layers



- neurons in the hidden layer:



$$y_k^2 = f^2\left(b_k^2 + \sum_{j=1}^{J}\left(w_{jk}^2 \times y_j^1\right)\right)$$

The functional looks as:

$$y = b^3 + \sum_{k=1}^{K}\left(w_k^3 \times \underbrace{f^2\left(b_k^2 + \sum_{j=1}^{J}\left(w_{jk}^2 \times \underbrace{f^1\left(b_j^1 + \sum_{i=1}^{I}\left(w_{ij}^1 \times x_i\right)\right)}_{y_j^1:\ \text{output of first hidden layer}}\right)\right)}_{y_k^2:\ \text{output of second hidden layer}}\right)$$

The fitting algorithm of NN potential

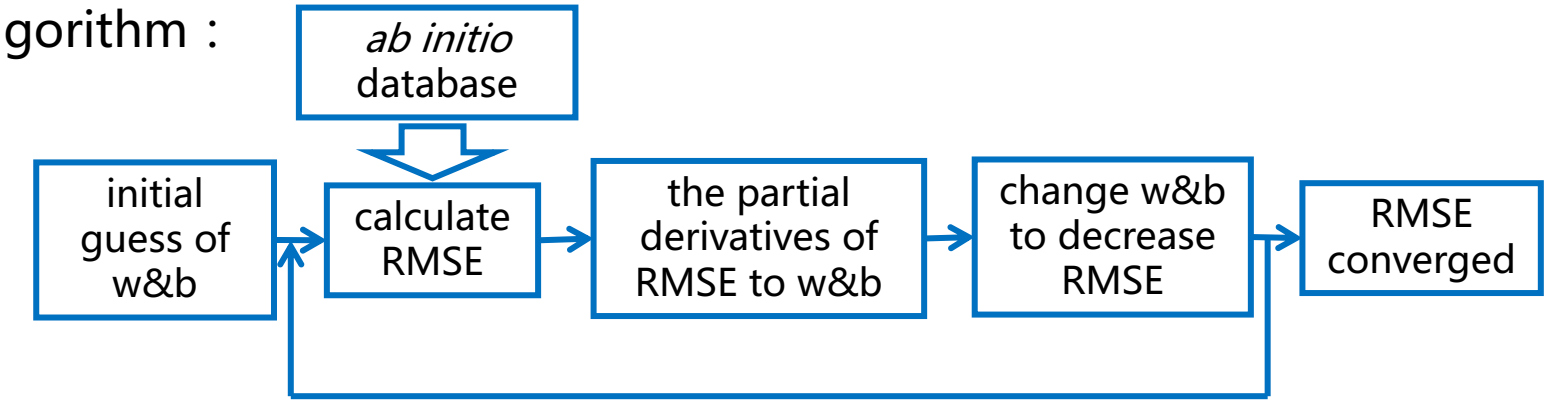
Loss function :$\mathrm{RMSE} = \sqrt{\frac{1}{Q}\cdot\sum_{q=1}^{Q}\left(E_q^{\mathrm{fit}} - E_q^{ab\ initio}\right)^2}$

$$y = b^3 + \sum_{k=1}^{K}\left(w_k^3 \times f^2\left(b_k^2 + \sum_{j=1}^{J}\left(w_{jk}^2 \times \underbrace{f^1\left(b_j^1 + \sum_{i=1}^{I}\left(w_{ij}^1 \times x_i\right)\right)}_{y_j^1:\ \text{output of first hidden layer}}\right)\right)\right)$$

y_k^2: output of second hidden layer

Neural network fitting: choose an appropriate number of hidden layers and neurons, and adjust the network parameters to minimize the error function.

Fitting algorithm :



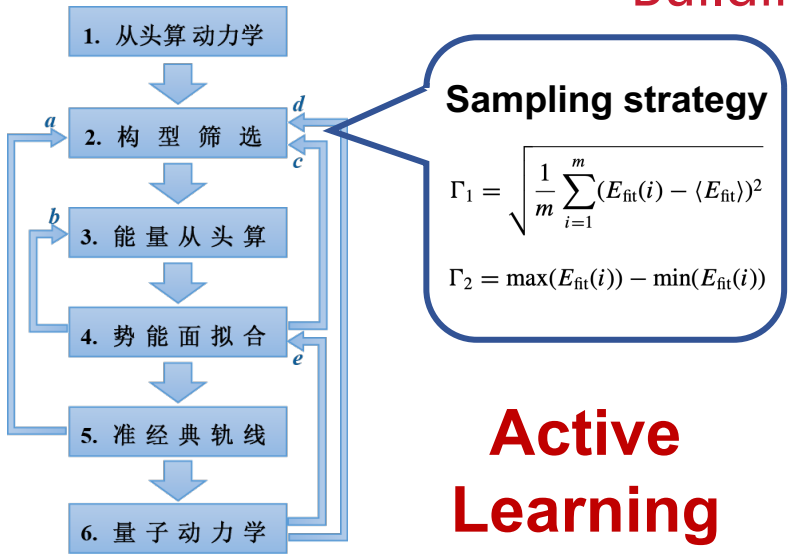
The Levenberg-Marquardt algorithm:

Hagan and Menhaj, *IEEE Trans. Neural Netw.*, **5**, 989 (1994)

practical section this afternoon!

Current bottlenecks of high-precision PESs:

Achieving spectroscopic-level accuracy in fitting;
Building datasets that surpass GGA-level quality.

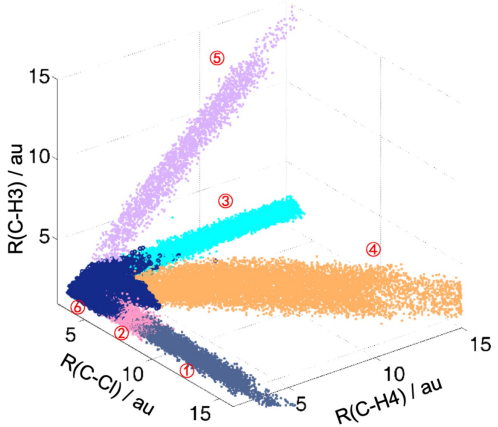


Sampling strategy

$$\Gamma_1 = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(E_{\mathrm{fit}}(i) - \langle E_{\mathrm{fit}}\rangle)^2}$$

$$\Gamma_2 = \max(E_{\mathrm{fit}}(i)) - \min(E_{\mathrm{fit}}(i))$$

Active Learning



Segmental Regression

$$E_{\mathrm{total}} = E\left(\frac{\mathrm{UCCSD(T)}}{\mathrm{AVQZ}}\right) \qquad \text{Spline}$$

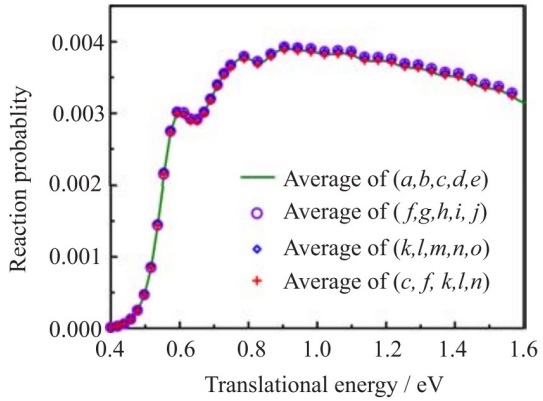
$$+\left[E\left(\frac{\mathrm{UCCSD(T)}}{\mathrm{AV5Z}}\right) - E\left(\frac{\mathrm{UCCSD(T)}}{\mathrm{AVQZ}}\right)\right] \qquad \text{1882 points + NN fitting}$$

$$+\left[E\left(\frac{\mathrm{UCCSD(T)}}{\mathrm{AV6Z}}\right) - E\left(\frac{\mathrm{UCCSD(T)}}{\mathrm{AV5Z}}\right)\right]\times\frac{5^{-3}}{5^{-3}-6^{-3}} \qquad \text{1184 points + NN fitting}$$

$$+\left[E\left(\frac{\mathrm{UCCSDT(2)_Q}}{\mathrm{AVQZ_F/VQZ_H}}\right) - E\left(\frac{\mathrm{UCCSD(T)}}{\mathrm{AVQZ_F/VQZ_H}}\right)\right] \qquad \text{775 points + NN fitting}$$

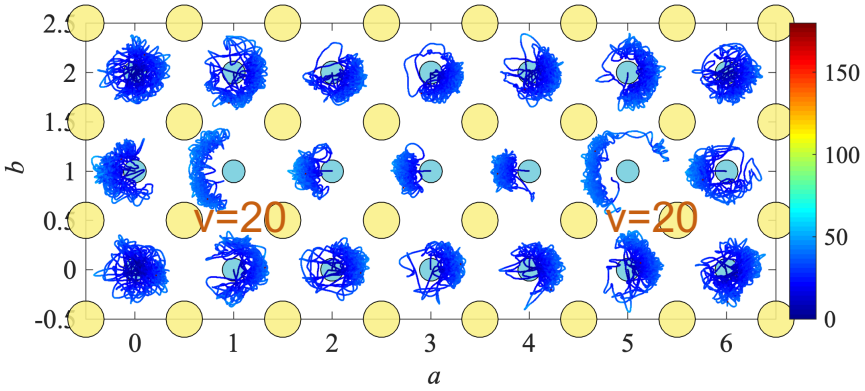
$$+E_{\mathrm{spin\text{-}orbit}} \qquad \text{Spline (Spin-orbit coupling)}$$

Hierarchical Scheme or Δ-ML



Averaging Ensemble

Many-body Expansion



- Chen, Xu, Xu, Zhang, ***J. Chem. Phys.***, 2013, 138, 154301
- Xu, Chen, Zhang, ***Chin. J. Chem. Phys.***, 2014, 27, 373
- Chen, Chen, Liu, Wu, Yang, Zhang, et al., ***PNAS***, 2020, 117, 9202
- Fu, Xu, Zhang, ***J. Chem. Phys.***, 2008, 129, 011103
- Wang, Chen, Yang, Zhang, et al., ***Science***, 2013, 342, 1499
- Chen, Li, Bowman, Guo, ***J. Phys. Chem. C***, 2020, 124, 19146

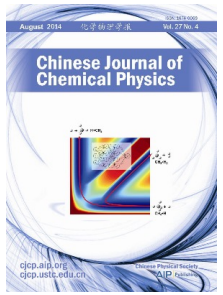
Neural network–based high-precision PESs

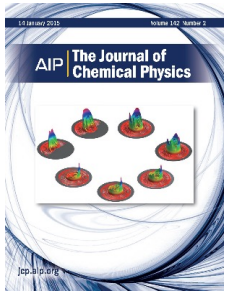
Gas phase reactions

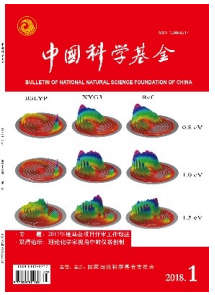
- FH_2: **Science 342** 1499
- ClH_2: **Science 347** 60
- OH_3: JCP **138** 154301
- HOCO: JCP **140** 044327
- CH_3F: JCP **144** 204302
- CH_5: JCP **142** 204302

Molecule @ surfaces

- HCl/Au(111): **JCP 139** 184705
- H_2O/Ni(100): **JCP 148** 144705
- H_2O/Cu(111): **PCCP 18** 26358
- CH_4/Ni(111): **PCCP 19** 30540
- ……

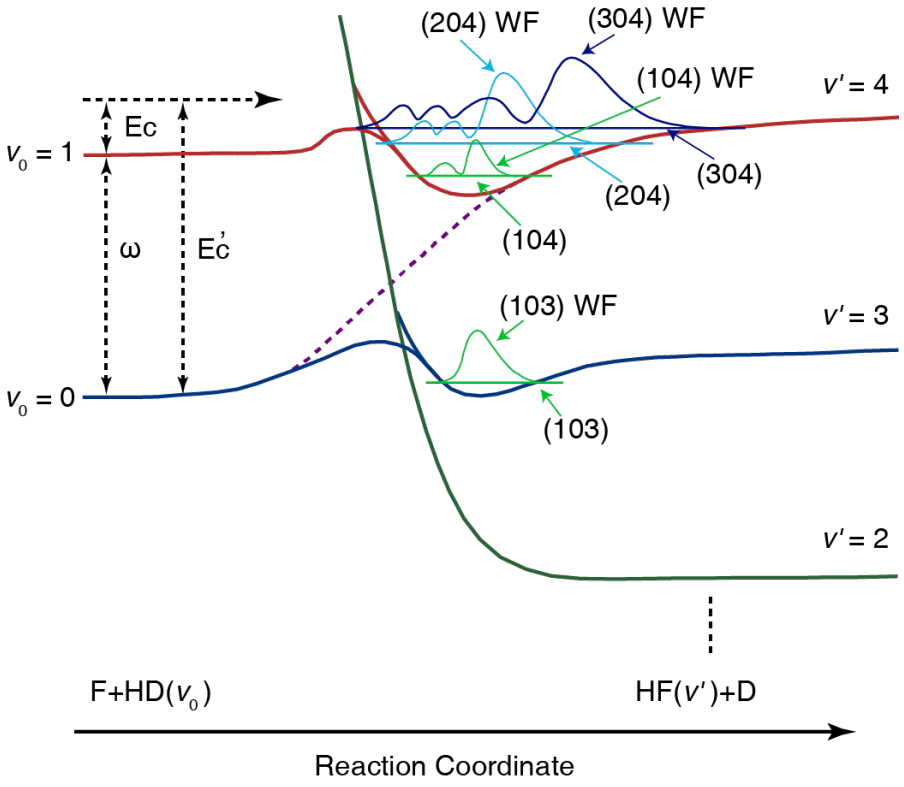


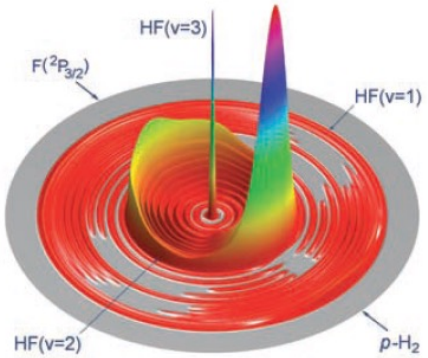




As research delves deeper, increasingly precise theoretical calculations are required to explain experimental phenomena.

evolution of the $F+H_2$ PES



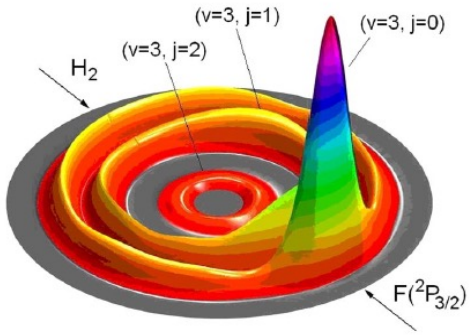


Science, 2006

forward scattering of HF(v'=2)

XXZ: MRCI(7e,9o)

15000 spline

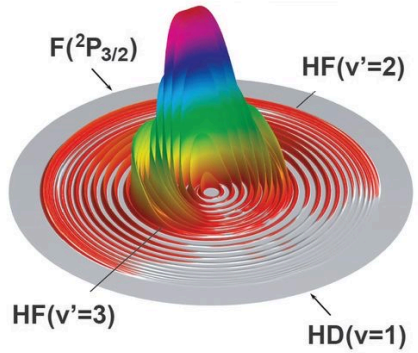


PNAS, 2008

forward scattering of HF(v'=3)

FXZ: CCSD(T)

15000 spline



Science, 2013

resonance state of F+HD(v=1)

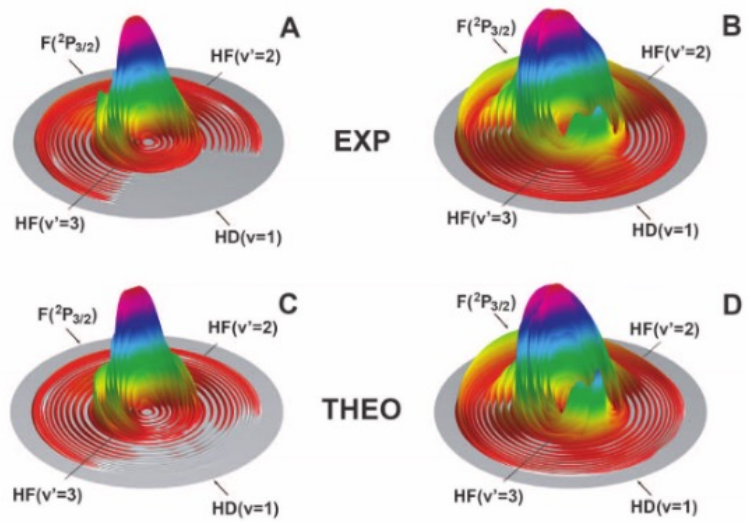
CSZ: CCSDT(Q)

775 NN

The number of *ab initio* calculations increase as the system size grows

$F+H_2$

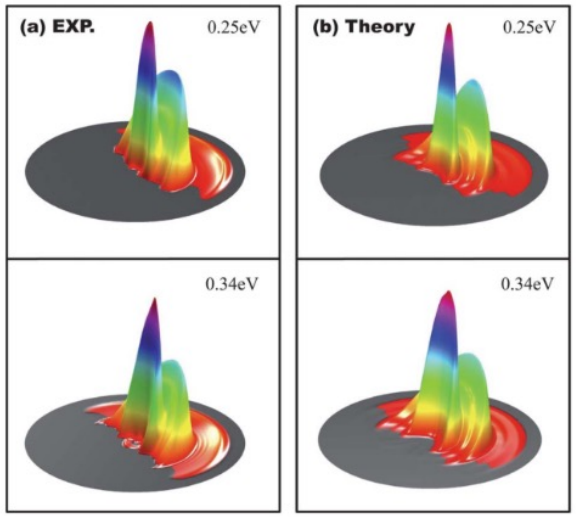
775 CCSDT(Q)/AVQZ



Chen-Sun-Zhang, JCP 2015

$H+H_2O$

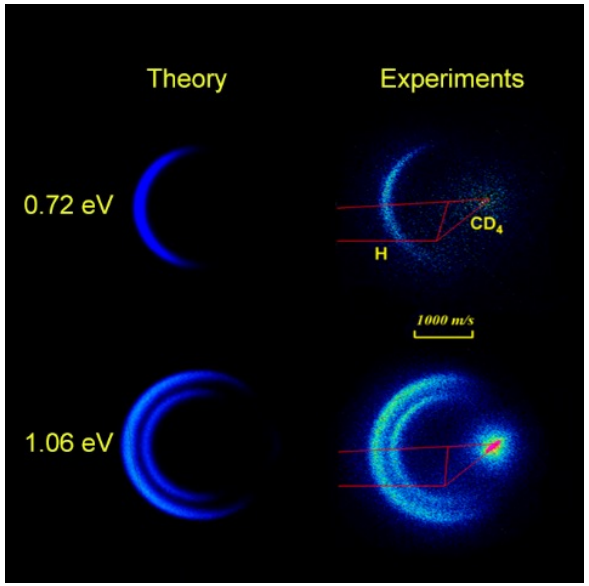
16814 CCSD(T)/AVTZ



Chen-Xu-Xu-Zhang, JCP 2013

$H+CH_4$

47783 CCSD(T)/AVTZ



Xu-Chen-Zhang, CJCP 2014

practical section this afternoon!

Thank You!

Prof. Dr. Jun Chen
Email: chenjun@fjirsm.ac.cn / njuchenjun@gmail.com
https://orcid.org/0000-0002-8021-7458
https://scholar.google.com/citations?user=zbCkt7gAAAAJ