MACHINE LEARNING BASED BAND GAP SCREENING FOR ULTRATHIN MATERIALS

M.Sc. Thesis

By

VARDHMAN DWIVEDI

DEPARTMENT OF PHYSICS

INDIAN INSTITUTE OF TECHNOLOGY INDORE

JUNE 2021

MACHINE LEARNING BASED BAND GAP SCREENING FOR ULTRATHIN MATERIALS

A THESIS

Submitted in partial fulfillment of the
requirements for the award of the degree
of
Master of Science

by

VARDHMAN DWIVEDI

DEPARTMENT OF PHYSICS
INDIAN INSTITUTE OF TECHNOLOGY INDORE

JUNE 2021

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **"MACHINE LEARNING BASED BAND GAP SCREENING FOR ULRATHIN MATERIALS"** in the partial fulfillment of the requirements for the award of the degree of **MASTER OF SCIENCE** and submitted in the **DEPARTMENT OF PHYSICS , Indian Institute of Technology Indore**, is an authentic record of my own work carried out during the time period from July 2019 to June 2021 under the supervision of Dr. Sudip Chakraborty, Assistant Professor.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Dwivedi
14/06/2021

Signature of the student with date
VARDHMAN DWIVEDI

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Sudip Chakraborty.

Signature of thesis Supervisor
Dr. Sudip Chakraborty
Department of Physics
Indian Institute of Technology, Indore
Indore-453552, India

डॉ. पंकज र. सगदेव/Dr. Pankaj R. Sagdeo
विभाग प्रमुख/Head
भौतिकी शास्त्र विभाग/Department of Physics
भारतीय प्रौद्योगिकी संस्थान इन्दौर
Indian Institute of Technology Indore

Signature of Administrative Supervisor
Dr. Pankaj R. Sagdeo
Department of Physics
Indian Institute of Technology, Indore
Indore-453552, India

VARDHMAN DWIVEDI has successfully given his/her M.Sc. Oral Examination held on **18th June 2021**.

Sudip Chakraborty.

डॉ. पंकज र. सगदेव / Dr. Pankaj R. Sagdeo
विभाग प्रमुख / Head
भौतिकी शास्त्र विभाग / Department of Physics
भारतीय प्रौद्योगिकी संस्थान इन्दौर
Indian Institute of Technology Indore

Signature of Thesis Supervisor
Date: 20/6/2021

Signature of Administrative Supervisor
Date: 22.06.2021

24-06-2021

Convener, DPGC
Date:

Signature of PSPC Member 1
Date: 20/6/2021

Signature of PSPC Member 2
Date: 21-06-2021

Signature of PSPC Member 3
Date: 20 JUNE 2021

ACKNOWLEDGEMENTS

Through this writing, I would like to extend my gratitude to all those who have helped me and shared with me their knowledge and wisdom.

This thesis is a result of dedicated guidance from my supervisor Dr. Sudip Chakraborty, Assistant professor. I would like to express my deeply-felt thanks and sincere gratitude to him, for introducing me to the field of research and for his constant assistance and inspiration. I am thankful to the senior PhD scholars of MATES group and to the department of Physics, Indian Institute of Technology, Indore for supporting during the course of this study.

VARDHMAN DWIVEDI

Abstract

The basics of Application of **Machine Learning** in **Density functional Theory** (DFT) to help us in electronic structure calculations for ultrathin 2D materials, and the quest of new ultrathin 2D materials have become a new field of research. To explain how we are approaching and progressing in this project and what will be the motivation, we will start by drawing an analogy. Let us suppose that, this DFT is my car, and we need to learn how to drive a car. So, to start with this, I will first get myself acquainted with the working principle of the car. For instance, the role of engine, gear box, shaft, handbrakes inside my car. Until we do not know these stuffs, we would not be able to understand why we drive the car in lower gear on steep hills and on higher gear on a flat road. So, before we start doing electronic calculations for different materials, we must know the functioning and the physics phenomena working behind DFT. So, the first phase of our project was to understand the working principle of DFT. The second phase is to drive the car, under the supervision of an experienced and learned person who knows driving. And to indentify what modifications are needed to be done in my car to optimize it. In this case, right now we are studying the things which can modify my car aka. DFT and getting used to the tools required to add those things in my car, such as programming/coding in different languages (for example: c++ and Python). That thing is called Machine learning. The third phase is to make use of my car for different purposes. For instance, going to the office, or grocery store, or hospital. In our case, we will use DFT for calculating electronic structures of ultrathin 2D materials. And then we will think of a process to apply Machine Learning (ML) in my car (DFT) to speed up our car so that we can reach our desired location (where we use to go very often) quickly. ML methodologies have capability to speed up our electronic structure calculation. we will also think of a way to make use of ML methodologies to predict new ultrathin 2D materials as well. We have made a Machine Learning model, which takes the different energy band gaps of corresponding 2D materials, and have successfully screened them on the basis of band gap between 1.2 eV to 1.3 eV.

TABLE OF CONTENTS

LIST OF FIGURES:

CHAPTER 1: INTRODUCTION TO DENSITY FUNCTIONAL THEORY.

1.1 BASIC HAMILTONIAN FOR INTERACTING ELECTRONS AND NUCLIE:

$$\hat{\mathrm{H}} = -\sum_{i=1}^{N} \frac{1}{2m_e} \nabla_i^2 - \sum_{I=1}^{M} \frac{1}{2\mathrm{M_i}} \nabla_I^2 - \sum_{i=1}^{N} \sum_{I=1}^{M} \frac{Z_I}{\left|\vec{r}_i - \vec{R}_I\right|} + \frac{1}{2} \sum_{i=1}^{N} \sum_{i \neq j}^{N} \frac{1}{\left|\vec{r}_i - \vec{r}_j\right|} + \frac{1}{2} \sum_{I=1}^{M} \sum_{I \neq J}^{M} \frac{\vec{z}_I \vec{z}_J}{\left|\vec{R}_I - \vec{R}_J\right|}$$

This is a Hamiltonian ($\hat{\mathrm{H}}$) for many-body systems where 'm_e' is mass of an electron, '$\mathrm{M_i}$' is mass of nucleus, '$\vec{r}$' is position vector of electrons and '$\vec{R}$' is position vector of nucleus, '$\vec{z}$'is the charge associated to the nuclei and M & N are the number of electrons and ions. First term represent the kinetic energy associated to the electron, second term represent kinetic energy associated to the nuclei, third term represent the interaction between electron and the nuclei/ion, fourth term is inter electronic interaction, and the last term is inter-nuclear interaction. The Hamiltonian in use is very complex and therefore extracting any experimental data from this Hamiltonian is impossible. So, we use an assumption called Born Oppenheimer (or adiabatic) approximation which states that the nuclear motion and the electronic motion in molecules can be separated. According to this approximation, the nuclear motion is so much slower than the electron motion that nucleus can be considered to be fixed and their kinetic energies can be neglected. This is because, the approximation in picture assumes the mass of nucleus is very very greater than the mass of an electron, therefore nucleus are going to move much slowly than electrons. Hence Kinetic energies of nucleus and inter-nuclear interaction are neglected. So, we are going to consider that position vector of nucleus ($\vec{R}$) is fixed. $\vec{R}$ is assumed to be parameters which reduced complexity of the system and interaction between nuclei is going to be constant.

$$\hat{\mathrm{H}}\,\Psi_K\left(\vec{r}_1\sigma_1, \vec{r}_2\sigma_2 \ldots \vec{r}_N\sigma_N \ldots \vec{R}_1, \vec{R}_2 \ldots \vec{R}_N\right) = E\Psi_K\left(\vec{r}_1\sigma_1, \vec{r}_2\sigma_2 \ldots \vec{r}_N\sigma_N \ldots \vec{R}_1, \vec{R}_2 \ldots \vec{R}_N\right)$$

Our aim is to characterize the eigen state Ψ_K and eigen value E of the Hamiltonian we have here. This is eigen equation of many body systems where 'σ' is spin states of that corresponding electron. Since **electrons are fermions**, so the wave-function we have to deal with needs to be **anti-symmetric**.

$$\Psi\left(\vec{r}_1\sigma_1 \ldots \vec{r}_i\sigma_i \ldots \vec{r}_j\sigma_j \ldots \vec{r}_N\sigma_N\right) = -\Psi\left(\vec{r}_1\sigma_1 \ \ldots \vec{r}_j\sigma_j \ldots \vec{r}_i\sigma_i \ldots \vec{r}_N\sigma_N\right)$$

&

$$\mathrm{N} \approx 10^{24} e^- / cm^3$$

And number of electron we have in a solid is very large. This is extremely complex. Therefore, It is impossible to have accurate wave function or eigen value for the above eigen equation of the system. All the calculation has to be based on variational methods (of quantum mechanics) or approximations. [1]

1.2 SYMMETRIC AND ANTI-SYMMETRIC WAVE FUNCTION:

Let us suppose we have a system with two electrons (a helium atom). Both the electrons occupy different quantum states Ψ_A & Ψ_B. Eigen states will determine the energy in which electrons are present and they are not dependent of $\vec{r}_1, \vec{r}_2$ here. Because, for the first electron, the eigen state is Ψ_A, then the position will determine the probability distribution of the electron and the probability distribution will only give a range of different values of r in which the electron can be present. Therefore, electrons of eigen state Ψ_A can be present in N number of locations. Ψ determines eigen state, which determines energy levels.

1.2.3 Combined wave functions solution:

$$\text{First electron}: \quad \vec{r}_1 \rightarrow \Psi_A$$
$$\text{Second electron}: \vec{r}_2 \rightarrow \Psi_B$$

$\Psi\left(\vec{r}_1, \vec{r}_2\right) = \Psi_A\left(\vec{r}_1\right) \ \Psi_B\left(\vec{r}_2\right)$ ---(1), electrons are identical in nature therefore probability distribution for exchanged system and probability for previous system will exactly be the same. Suppose both these two electrons are at two distinct energy levels, what if they suddenly swap places and the electron in one of the energy level changes place with electron in another energy level ? There is not going to be any overall change in the configuration in energy level and in the probability distribution either.

1.2.4 Combined wave function solution after exchange of particles:

After the exchange of particle:

First electron: $\vec{r}_2 \rightarrow \Psi_A$

Second electron: $\vec{r}_1 \rightarrow \Psi_B$

-

$$\Psi(\vec{r}_2, \vec{r}_1) = \Psi_A(\vec{r}_2)\ \Psi_B(\vec{r}_1) \quad \cdots(2)$$

$$\left|\Psi(\vec{r}_1, \vec{r}_2)\right|^2 = \left|\Psi(\vec{r}_2, \vec{r}_1)\right|^2$$

Probability distribution is invariant under exchange of electrons. This leads to fundamental classification of elementary particle.

$$\Rightarrow \Psi(\vec{r}_1, \vec{r}_2) = \pm\Psi(\vec{r}_2, \vec{r}_1)$$

1.Symmetric (BOSONS) : -

$$\Psi(\vec{r}_1, \vec{r}_2) = \Psi(\vec{r}_2, \vec{r}_1)$$

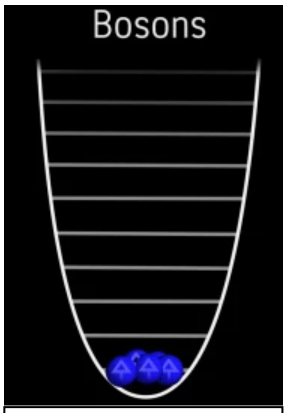


Fig 1.1: Bosons

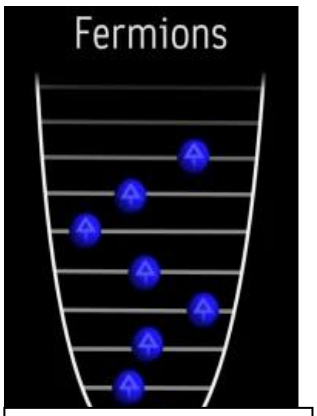


Fig 1.2: Fermion

2.Anti - symmetric (FERMIONS) : -

$$\Psi(\vec{r}_1, \vec{r}_2) = -\Psi(\vec{r}_2, \vec{r}_1)$$

Both these two wave function solutions ((1) & (2)) are possible if you are making measurement of system consisting two particles and both the two particle underwent some kind of exchange, you would not know from your measurement. So, both these wave function solutions are possible. Linear combination of both the equation (1) and (2) gives us the general solution of the two electron system.

$$\Psi_{\text{sym}} = \frac{1}{\sqrt{2}}\left[\Psi(\vec{r}_1,\vec{r}_2)+\Psi(\vec{r}_2,\vec{r}_1)\right];\quad \Psi_{antisym} = \frac{1}{\sqrt{2}}\left[\Psi(\vec{r}_1,\vec{r}_2)-\Psi(\vec{r}_2,\vec{r}_1)\right]$$

Special case: Both the particle exist in same quantum state :-

First electron : $\vec{r}_1 \rightarrow \Psi_A$

Second electron : $\vec{r}_2 \rightarrow \Psi_A$

Combined wave function : -

$\Psi(\vec{r}_1,\vec{r}_2) = \Psi_A(\vec{r}_1)\ \Psi_B(\vec{r}_2)$

$\Psi(\vec{r}_2,\vec{r}_1) = \Psi_A(\vec{r}_2)\ \Psi_B(\vec{r}_1)$

1.Symmetric wavefunction : -

Ψ_{sym} = some value

2.Anti - symmetric wavefunction : -

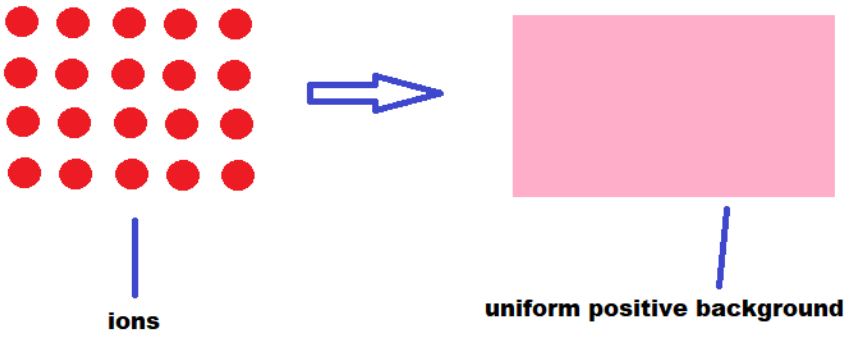
$\Psi_{\text{antisym}} = 0$ (vanishes)

which is not possible!!

Not possible as it implies that these two particles simply do not exist at all.. But electrons are existing somewhere, therefore there must be some probability distribution, otherwise, how does it vanish. The two particle cannot exist in the same given quantum state, this leads to **pauli-exculsion principle.** For certain class of particles, called fermions, which follow anti-symmetric wave function condition, it is not possible for them to exist in same quantum state, but it is possible for bosons. [2

1.3 UNIFORM ELECTRON GAS SYSTEM:

(Example of simple approximation)



$$\rho = \frac{N}{V}$$

Fig 1.3: Uniform electron gas system

ρ is ionic charge density. First step in this approximation is substituting all ions and nucleus into uniform positive background. Hamiltonian of this gas is :-

$$\hat{H} = -\sum_{i=1}^{N} \frac{1}{2m_e} \nabla_i^2 + \frac{1}{2} \sum_{i=1}^{N} \sum_{i \neq j}^{N} \frac{1}{|\vec{r}_i - \vec{r}_j|} \equiv H_{e\,I}$$

$$-\sum_{i=1}^{N} \int d^3\vec{R} \frac{\rho}{|\vec{r}_i - \vec{R}|} \equiv H_{e\text{-}i}$$

$$+\frac{1}{2} \int\int d^3\vec{R} d^3\vec{R}' \frac{\rho^2}{|\vec{R} - \vec{R}'|} \equiv H_{i\text{-}i}$$

Where is $H_{e\,I}$ inter electronic interaction, is $H_{e\text{-}i}$ electrons and ions interaction and is $H_{i\text{-}i}$ ion-ion interaction. If electrons are also distributed, then, $n = \rho$, Where n **is electronic charge density,** then our Hamiltonian becomes :-

$$\hat{H} = -\sum_{i=1}^{N} \frac{1}{2m_e} \nabla_i^2 + \frac{1}{2} \int\int d^3\vec{r} d^3\vec{r}' \frac{n^2}{|\vec{r} - \vec{r}'|} - \int\int d^3\vec{r} d^3\vec{R} \frac{n^2}{|\vec{r}_i - \vec{R}|}$$

$$+\frac{1}{2} \int\int d^3\vec{R} d^3\vec{R}' \frac{n^2}{|\vec{R} - \vec{R}'|}$$

All the contribution in interaction potential are cancelled and only term left is Kinetic energy associated to electron (shaded in blue). Charge density of uniform positive background is equal to ρ, which is equal to electronic charge density, so as to maintain neutrality, $\rho = \frac{N}{V}$, where N is number of electron.[3]

1.4 HARTREE APPROXIMATION:

Now, free electron gas approximation was not enough to analyze physical properties of solid. First approximation we consider is Hartree approximation.

$$\hat{H}\Psi = \sum_{i=1}^{N}\left(-\frac{1}{2m_e}\nabla_i^2 - \sum_{I=1}^{M}\frac{Z_I}{\left|\vec{r}_i - \vec{R}_I\right|}\right)\Psi + \frac{1}{2}\sum_{i=1}^{N}\sum_{i\neq j}^{N}\frac{1}{\left|\vec{r}_i - \vec{r}_j\right|}\Psi = E\Psi$$

First term in the above equation is Kinetic energy associated with electrons, second term is electron-ion interaction and last term is electron-electron interaction. First goal is to get eigen states and eigen values of the Hamiltonian.

1.4.1 Hartree Approximation condition:

a) $\Psi_K(\vec{r}_1\sigma_1, \vec{r}_2\sigma_2 ... \vec{r}_N\sigma_N) = \psi_1(r_1,\sigma_1)\psi_{21}(r_{21},\sigma_2)....\psi_N(r_{N1},\sigma_N)$

Independent electron picture but No anti-symmetry.

+

b) Variational principle $\Rightarrow$ Minimize $\left\langle \hat{H} \right\rangle_\Psi = \left\langle \Psi \left| \hat{H} \right| \Psi \right\rangle$

$$\left\langle \Psi | \varphi \right\rangle = \sum_{\sigma_1,..\sigma_N}\int d^3\vec{r}_1 ... d^3\vec{r}_N \Psi^*(\vec{r}_1\sigma_1,....\vec{r}_N\sigma_N)\varphi(\vec{r}_1\sigma_1,....\vec{r}_N\sigma_N)$$

(a). Within this Hamiltonian, we cannot apply independent electron picture, since the last term cannot be considered independent. But, considering that within this approximation, we are looking for some kind of **effective potential** that each electron is feeling, this effective potential could be affected by all the rest of the electrons and considering that effective potential, the electron could be moving like indefinitely for the others. This is the idea behind Hartree-Approximation. And considering independent electron picture, it means that

the wave function we are going to try here is the product of the wave functions of the individual electron.

(b). Variational Approximation:

If we cannot find an analytic solution to the Schrödinger equation, this trick allows us to estimate the energy of the ground state of a system.

$$\langle \Psi | \varphi \rangle = \sum_{\sigma_1,...\sigma_N} \int d^3\vec{r}_1 ... d^3\vec{r}_N \Psi^*(\vec{r}_1\sigma_1,....\vec{r}_N\sigma_N)\varphi(\vec{r}_1\sigma_1,....\vec{r}_N\sigma_N)$$

We are looking for stationary states (or eigen state with all observable independent of time) of the Hamiltonian of unconstrained wave function that is wave function of multi-particle Schrödinger equation. Now, first step is to calculate the average Hamiltonian considering wave function.

$$\left\langle \hat{\mathrm{H}} \right\rangle_\Psi = \sum_\sigma \sum_{i=1}^{N} \int d^3\vec{r}\, \psi^*_{i(\vec{r},\sigma)} \left(-\frac{1}{2m_e}\nabla_i^2 + u_{ion}(\vec{r}) \right) \psi_{i(\vec{r},\sigma)}$$

$$+ \frac{1}{2} \sum_{(j,\sigma')\neq(i,\sigma)} \int\int d^3\vec{r} d^3\vec{r}' \frac{1}{|\vec{r}-\vec{r}'|} |\psi_j(\vec{r}',\sigma')|^2 |\psi_i(\vec{r},\sigma)|^2$$

$$and \quad \sum_i \sum_\sigma |\psi_i(\vec{r},\sigma)|^2 = n(\vec{r})$$

Basically, what we have here in the last term of average Hamiltonian is the interaction energy, which means, whenever we have a system which is characterized by a charge density and R, the interaction potential of a system or its electron density is given by this expression. Now to avoid self interaction, that is electron A cannot interact with itself, we have $i \neq j$, therefore we have to minimize that contribution or that expression. But wave function we are going to use here have a constraint, they have to be normalized. Basically, it means that the wave function for each electron has to be normalized.

$$\sum_\sigma \int d^3\vec{r} |\psi_i(\vec{r},\sigma)|^2 = 1$$

Whenever we are doing minimization with constraints, we have to use Lagrange multipliers method. We have to define a new function,

using lagrange multipliers $\Downarrow$

$$\mathrm{F}\left[\psi_i(\vec{r},\sigma),\varepsilon_i\right]=\left\langle\hat{\mathrm{H}}\right\rangle_{\Psi}-\sum_i\varepsilon_i\left(\sum_{\sigma}\int d^3\vec{r}\left|\psi_i(\vec{r},\sigma)\right|^2-1\right)$$

Now, we will do minimization with respect to Lagrange multipliers (used to find local maxima and minima subject to equality constraint), therefore doing minimization:-

$\frac{\delta \mathrm{F}}{\delta\psi}=0$ Functional derivatives taking $\delta\psi$ and $\delta\psi^*$ as independent variation

Functional derivative can be done by taking any of the $\psi_{i(\vec{r},\sigma)}$ or $\psi^*{}_{i(\vec{r},\sigma)}$

Hartree equation $\Rightarrow$

$$-\frac{1}{2m_e}\nabla^2\psi_i(\vec{r},\sigma)+u_{ion}\psi_i(\vec{r},\sigma)+\sum_{(j,\sigma')\neq(i,\sigma)}\int d^3\vec{r}'\frac{\left|\psi_j(\vec{r}',\sigma')\right|^2}{\left|\vec{r}-\vec{r}'\right|}\psi_i(\vec{r},\sigma)=\varepsilon_i\psi_i(\vec{r},\sigma)$$

We need to solve the second term of left and side of the equation. This term is giving us electronic charge density so this is like interaction potential that electron 'i' can see and can cast or induced by all the rest of the electrons we have in the system. This is effective potential that electron 'i' can see and the system encamped by all the rest of the electrons. This effective potential here is doing the screening. All the electrons are screening the potential created by the ions, which is complicated to solve. Now we start with initial guess for all the wave functions $\psi^0{}_{i(\vec{r},\sigma)}$ and we plug it in $\psi_{j(\vec{r},\sigma)}$ and solving it, we get a new $\psi_{i(\vec{r},\sigma)}$ and use it again.[4]

1.5 HARTREE-FOCK APPROXIMATION:

This time, we change one thing, that is, we considered wave function to be anti-symmetric, and for describing such system, we will have to use **Slater determinant.** It is an expression that describes the wave function of a multi-fermionic system. It satisfies anti-symmetric requirements.

$$a)\ \Psi(\vec{r}_1\sigma_1....\vec{r}_N\sigma_N) = \frac{1}{\sqrt{N!}}\begin{vmatrix} \psi_1(\vec{r}_1,\sigma_1) & \psi_1(\vec{r}_2,\sigma_2) & \cdots & \psi_1(\vec{r}_N,\sigma_N) \\ \psi_2(\vec{r}_1,\sigma_1) & \psi_2(\vec{r}_2,\sigma_2) & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \psi_N(\vec{r}_1,\sigma_1) & \psi_N(\vec{r}_2,\sigma_2) & \cdots & \psi_N(\vec{r}_N,\sigma_N) \end{vmatrix}$$

(Independent electron picture and anti - symmetry)

b) Variational approximation $\Rightarrow$ Minimize $\left\langle \hat{H} \right\rangle_\Psi = \left\langle \Psi \left| \hat{H} \right| \Psi \right\rangle$

$$\left\langle \hat{H} \right\rangle = \textbf{K.E term} + e-ion \textbf{ interaction term} + e-e \textbf{ interaction term}$$

$$-\frac{1}{2}\sum_{j,\sigma'}\int d^3\vec{r}\,'\frac{1}{|\vec{r}-\vec{r}\,'|}\delta_{\sigma,\sigma'}\psi_i^*(\vec{r},\sigma)\psi_i(\vec{r}\,',\sigma')\psi_j^*(\vec{r}\,',\sigma')\psi_j(\vec{r},\sigma)$$

$$F[\psi_i(\vec{r},\sigma),\varepsilon_i] = \left\langle \hat{H} \right\rangle_\Psi - \sum_i \varepsilon_i\left(\sum_\sigma \int d^3\vec{r}\,|\psi_i(\vec{r},\sigma)|^2 - 1\right)$$

The last term of average Hamiltonian given above is the exchange term due to anti-symmetric. This third contribution exists if spin states of 'i' and 'j' states are same. Therefore, having same state and negative sign, so it is lower in the energy, basically it means, it is favouring the states having same spin. Therefore it favour ferromagnetic state, so spins are aligned.

$$\text{Hartree fock equation}: \quad \frac{\delta F}{\delta \psi} = 0$$

same terms as in hartree equation

$$-\sum_{j,\sigma'}\int d^3\vec{r}\,'\frac{1}{|\vec{r}-\vec{r}\,'|}\delta_{\sigma,\sigma'}\psi_j^*(\vec{r}\,',\sigma')\psi_i(\vec{r}\,',\sigma)\psi_j(\vec{r},\sigma') = \varepsilon_i\psi_i(\vec{r},\sigma)$$

Now, we can see that this Hartree Fock equation is not corresponding to solving eigen value and eigen state. They look like Hamiltonian but this does not look like equation for solving wave function and eigen value, so what actually does this Lagrange multiplier physical significance is? Answer comes from Koopmans equation.

$$\acute{\varepsilon}_K = E(N) - E(N-1)$$

$\acute{\varepsilon}_K$ is energy difference between the states with N electron and system with N-1 electrons. Considering that electron we are removing is in state ψ_K. [5]

1.6 THOMAS-FERMI MODEL:

This model includes electron density as key ingredient to solve the problem. Semi classical approximation for electron in an external potential, V_{ext} i.e., Quantum calculations with classical approximation. They thought instead of looking for wave function, it could be more efficient to look for directly electron charge density.

$$E(n) = \frac{3}{10}\left(3\pi^2\right)^{2/3}\int d^3\vec{r}\, n^{5/3}(\vec{r}) + \frac{1}{2}\int\int d^3\vec{r}\, d^3\vec{r}\,' \frac{n(\vec{r})n(\vec{r}\,')}{\left|\vec{r}-\vec{r}\,'\right|} + \int d^3\vec{r}\, V_{ext}\, n(\vec{r})$$

- The first expression on the right hand side of the above equation is based on uniform electron gas approximation.

- Expression for the total energy of the electron was:

$$\int \frac{3}{10}\left(3\pi^2\right)^{2/3} n^{2/3}$$

- And therefore the total energy is:

$$\int \frac{3}{10}\left(3\pi^2\right)^{2/3} n^{2/3}(\vec{r})n(\vec{r})d^3\vec{r}$$

- In uniform electron gas, the ‘n' was constant, but here it depends on (r). Hence,

V_{ext} is the potential experienced by the electron

$$\int \frac{3}{10}\left(3\pi^2\right)^{2/3} n^{2/3}(\vec{r})n(\vec{r})d^3\vec{r}$$

- Second term/expression is inter-electronic electrostatic potential. If electronic charge density is given by $n(\vec{r})$, then electrostatic charge potential associated with that charge distribution is given by this expression:

$$\frac{1}{2}\int\int d^3\vec{r}d^3\vec{r}'\frac{n(\vec{r})n(\vec{r}')}{\left|\vec{r}-\vec{r}'\right|}$$

- Third term is electrostatic potential associated with external potential. Now, we don't know the expression for $n(\vec{r})$, so how do we do that.

Total energy that the system is going to take is where it minimizes the energy of the system.

Search for $n(\vec{r})$ that minimizes $\mathrm{E}(n)$ subject to $\int n(\vec{r})d^3\vec{r} = N$

The last integral is the constraint (keeping the number of electron fixed).

Applying Lagrange multipliers:

$$\mathrm{F}(n) = \mathrm{E}(n) - \mu\left[\int n(\vec{r})d^3\vec{r} - N\right] \text{and} \frac{\delta \mathrm{F}}{\delta n} = 0$$

New functional is equal to the energy we defined before ' $-\mu$' times constraint.

Thomas - Fermi equation for $n(\vec{r})$ in integral form :

$$\frac{5}{3}\mathrm{A_K}n^{2/3}(\vec{r}) + \mathrm{V_{ext}}(\vec{r}) + \int d^3\vec{r}'\frac{n(\vec{r}')}{\left|\vec{r}-\vec{r}'\right|} = \mu \text{ where } \mu = \frac{\delta \mathrm{E}}{\delta n} \rightarrow \text{chemical potential}$$

$$\therefore \frac{\delta \mathrm{E}}{\delta n} - \mu = 0 \quad \text{and} \int d^3\vec{r}'\frac{n(\vec{r}')}{\left|\vec{r}-\vec{r}'\right|} = V$$ (related with poison's equation)

$\therefore \nabla^2 \mathrm{V}(\vec{r}) = 4\pi n(\vec{r})$ Hence, we can ultimately sum this up as:

Thomas - fermi eqn. for $n(\vec{r})$ in differerntial form :

$$V(\vec{r}) = \int d^3\vec{r}\,' \frac{n(\vec{r}\,')}{|\vec{r} - \vec{r}\,'|} \rightarrow \nabla^2 V(\vec{r}) = 4\pi n(\vec{r}) \text{[Poisson's eqn.]}$$

$$\text{From Thomas - Fermi eqn : } n(\vec{r}) = \left(\frac{3}{A_K}\right)^{3/2} (\mu + V_{ext}(\vec{r}) + V(\vec{r}))^{3/2}$$

$$\nabla^2 V(\vec{r}) = 4\pi \left(\frac{3}{A_K}\right)^{3/2} (\mu + V_{ext}(\vec{r}) + V(\vec{r}))^{3/2}$$

This is differential eqn. for $V(\vec{r})$

And once we calculate electron density, we can find energy of the system.

Pair Density and electron correlation:

$\hat{n}(\vec{r}\sigma, \vec{r}'\sigma')$

Joint probability $\Rightarrow$ of finding and electron of spin σ at point $\vec{r}$,
and another electron of spin σ' at point $\vec{r}'$.

$$\hat{n}(\vec{r}\sigma, \vec{r}'\sigma') = n(\vec{r}\sigma) n_2(\vec{r}\sigma, \vec{r}'\sigma')$$

$\downarrow$ $\downarrow$

probability of finding an electron of spin σ at point $\vec{r}$

conditional probability of finding the second electron of spin σ' at $\vec{r}'$

$\downarrow$

correlation

1.7 HOHENBERG-KOHN THEOREM:

This theorem established the basis of Density Functional Theory. DFT: considering electron density to be the key ingredients is determining the exact theory of many body system.

$$\hat{\mathrm{H}} = -\frac{1}{2m_e}\sum_i \nabla_i^2 + \underbrace{\sum_i \mathrm{V}_{\mathrm{ext}}(\vec{r}_i)}_{\mathrm{V}_{\mathrm{ext}}} + \underbrace{\frac{1}{2}\sum_{i=1}^{N}\sum_{i\neq j}^{N}\frac{1}{\left|\vec{r}_i - \vec{r}_j\right|}}_{\mathrm{V}_{\mathrm{e\text{-}e}}}$$

The approach: DFT as an exact theory of many body systems.

$$\begin{array}{ccc} \mathrm{V}_{\mathrm{ext}} & & n_0(\vec{r}) \\ \downarrow & & \uparrow \\ \Psi_i & \rightarrow & \Psi_0 \end{array}$$

If we explicitly determine $\mathrm{V}_{\mathrm{ext}}$, so we are précising our Hamiltonian, then, we could calculate eigen states Ψ_i of this Hamiltonian and with that we can calculate the ground state Ψ_0, and having the Ψ_0, we can have the electron density associated to that ground state.

Theorem 1: $\mathrm{V}_{\mathrm{ext}}$ is determined uniquely, except for constant, by $n_0(\vec{r})$ (the ground state particle density).

Corollary 1: Since the Hamiltonian is fully determined (expect for a constant shift for the energy) it follows that many body wave-functions for all the states (ground and excited). Therefore, all the properties of the system are completely determined given only the ground state density, $n_0(\vec{r})$.

The theorem states that if we have a ground state electron density, then this $n_0(\vec{r})$ uniquely determines $\mathrm{V}_{\mathrm{ext}}$, therefore we are determining the Hamiltonian, and once we have Hamiltonian, we can calculate ground state and eigen state. And once we have Ψ_i, we can calculate everything we want. If this Ψ_i is uniquely determined by this ground state $n_0(\vec{r})$, then, in principle, everything could be determined by this ground state electronic density. This is the main conclusion.

Theorem 2 : A universal functional for the energy E(n) in terms of the density $n(\vec{r})$ can be defined, for any $\mathrm{V}_{\mathrm{ext.}}$. The exact ground state energy of the system is the global minimum, and the density that minimizes this functional is the exact ground state density.

Corollary 2: E(n) alone is sufficient to determine the exact ground state energy and density. In general, excited state of electron must be determined by other means. [6]

1.8 KOHN-SHAM METHOD:

In the Hohenberg-Kohn Theorem, we saw that if we have this universal functional, we have this universal functional, we can get the ground state electronic density. Getting a universal functional is extremely difficult , there is no successful expression for that in terms of the density. Now the strategy which Kohn-Sham followed in order to have good expression for universal functional was this method. They divided or decomposed F(n) into two different terms.

Decomposition of universal functional

Faced with the difficulty of approximating directly F[n], kohn - sham :

$$F[n] = T[n] + E_{HXC}[n]$$

$T[n]$ is the non-interacting Kinetic energy functional (this is not the functional of kinetic energy of a real system). Idea was basically to substitute the real system with non-interacting system. And we know that the wave function of non-interacting system can be described by slater-determinants.

1).First term is the non - interacting kinetic energy functional defined by a constraint :

$$T[n] = \underbrace{\min}_{\Phi \to n} \langle \Phi | \hat{T} | \Phi \rangle = \langle \Phi[n] | \hat{T} | \Phi[n] \rangle$$

Kinetic energy is defined this way. For fixed electronic density, we minimize the kinetic energy of that non-interacting system, so that here, the wave function Φ we have are different slater determinant. So, we minimize Kinetic energy for different slater determinant that have the electronic density 'n'. This minimizing wave function is called Kohn-Sham wave function and is denoted by $\Phi[n]$. Of course it is going to be the functional of the density because, for fixed density, we minimize with respect to different slater determinants that give electron density.

$$T[n] = \underbrace{\min}_{\Phi \to n} \langle \Phi | \hat{T} | \Phi \rangle = \langle \Phi[n] | \hat{T} | \Phi[n] \rangle$$

$\downarrow$

Minimization is done over normalized slater - determinant wave function Φ which yields the fixed density *n*.

$\downarrow$

For a given density *n*, the minimizing slater – determinant wave fun. is called the KS wave fun. and is denoted by $\Phi[n]$

2). Second term is the hartree - exchange - correlation functional.

Decomposition of E_{HXC} functional:

$$E_{HXC}[n] = E_H[n] + E_{XC}[n]$$

$\downarrow$ $\downarrow$

This is hartree functional

Exchange-correlation energy functional

$\downarrow$

$E_{XC}[n] = E_X[n] + E_C[n]$: 1st terms is exchange functional while 2nd term is correlation

The advantage of Kohn-Sham method is that a major part of the kinetic energy can be treated explicitly with the single determinant wave function $\Phi[n]$ and only $E_{HXC}[n]$ needs to be approximated as a functional of the density.

1.8.1 Hohenberg-Kohn and Kohn-sham schematics:

H - K

$$V_{ext}(\vec{r}) \Leftarrow n_0(\vec{r})$$

$\Downarrow$ $\Uparrow$

$$\Psi_i(\{\vec{r}\}) \Rightarrow \Psi_0(\{\vec{r}\})$$

Once we specify $V_{ext}(\vec{r})$ that inter electronic system is feeling, we specify the total Hamiltonian. Once we have Hamiltonian H , we can calculate eigen states which is extremely difficult to calculate. And Hohenberg-Kohn Theorem states that this ground state wave function uniquely specifies the external potential. Therefore, ground state electron density $n_0(\vec{r})$ uniquely specifies total Hamiltonian H and then we can calculate $\Psi_0(\{\vec{r}\})$ According to Hohenberg-Kohn theorem, once we have the $n_0(\vec{r})$ of the system, we can get any other property of the system. In principle, we have to define a functional of the $n_0(\vec{r})$ in order to get any property of the system. Problem is that we don't know how to calculate this ground state density $n_0(\vec{r})$. This is extraordinarily difficult task because the system involves many interacting electrons. This is where Kohn-Sham method enters. Kohn-Sham proposed

following approach. The Kohn-Sham approach is to replace the original difficult interacting-particle Hamiltonian with a different one which could be solved more easily: a non-interacting particle Hamiltonian with an effective potential (v_s) assumed to have the same density as the true interacting system.

$$\begin{array}{ccccccc} & \text{H-K} & & \text{K-S} & & \text{H-K}_0 & \\ V_{ext}(\vec{r}) & \Leftarrow & n_0(\vec{r}) & \Leftrightarrow & n_0(\vec{r}) & \Rightarrow & v_s(\vec{r}) \\ \Downarrow & & \Uparrow & & \Uparrow & & \Downarrow \\ \Psi_i(\{\vec{r}\}) & \Rightarrow & \Psi_0(\{\vec{r}\}) & & \Phi & \Leftarrow & \phi_i(\vec{r})\chi_i(\vec{r}) \end{array}$$

In the Hohenberg-Kohn Theorem, everything could be determined as a function of the $n_0(\vec{r})$. For the same ground state electronic density, we have a non-interacting system. That is, we have defined a non-interacting system for the same $n_0(\vec{r})$. If we apply Hohenberg-Kohn theorem for the non-interacting system, then according to that, $n_0(\vec{r})$ uniquely specifies the external potential that each electron is feeling. Somehow, that is the effective potential associated to each electron. And this is what we call Kohn-Sham potential. And once we have Kohn-Sham potential, we can calculate any eigen state associated to that Hamiltonian { $\phi_i(\vec{r})\chi_i(\vec{r})$ }. Once we have this one particle wave function, we can deal therefore, with slater-determinant Φ of the N particle system in order to anti-symmetrizing those eigen states. Once we have Φ, that is going to be the wave function of this non-interacting system. Now, this Φ can give us ground state electron density, which is same as that obtained from Hohenberg-Kohn theorem for interacting system. Any property of this real system could be derived as a function of this slater determinant.

1.8.2 Kohn-Sham Method says:

1). In order to calculate ground state density, we are not going to solve the complex real system, but we are going to substitute which is non-interacting system, and it is much more easier to calculate. **AIM:** Aim is to get good approximation for that effective potential (that is Kohn-Sham Potential) in order to have Φ .[7], [8]

CHAPTER 2. MACHINE LEARNING:

2.1 Introduction to Machine Learning:

Humans have a tendency to gain an understanding from their past experiences and machines obeys the instructions given by humans. Now, what if humans can train the machines to learn from their past data and do what humans can do, much faster, that is , machine learning is about understanding and reasoning. Suppose, taking me as an example, and I love exploring music and listen new songs. I either like it or dislike it. I decide this on the basis of Tempo, Genre, Intensity

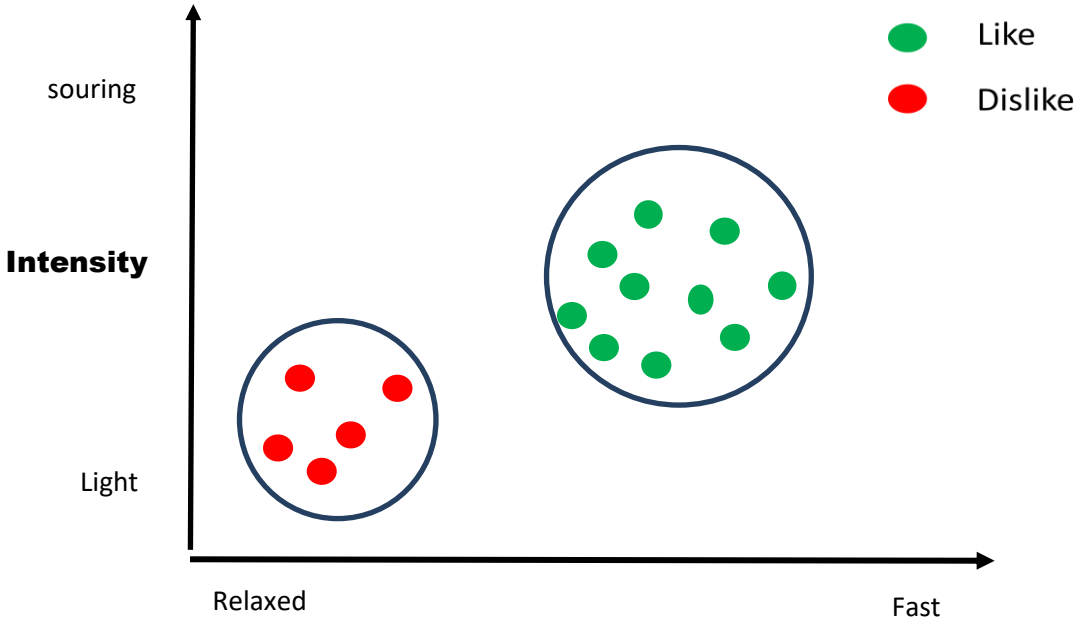


Fig 2.1: An example of Classification of data in machine

Now, you see that I like songs with fast tempo and soaring intensity, whereas, dislike the song with relaxed tempo and light intensity. Now, you know my choices. Now , suppose I listen to a new song, which has a fast tempo and a soaring intensity, therefore it will lie somewhere in the group with green dots.

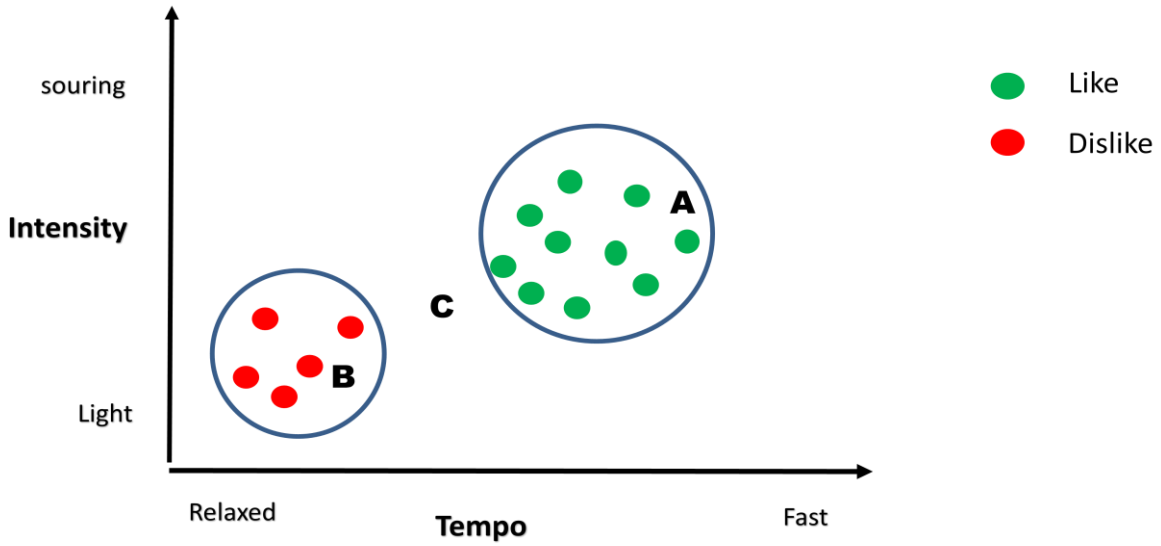


Fig 2.2 : A new data point is introduced

Now, looking where it lies, can you guess whether I would like the song or not? Yes, I would like the song. Looking at my past choices, you were able to classify the unknown song very easily. Now, suppose I listen to a song which lies somewhere between the two groups (as shown in the figure 5 above). Now, how do you classify, whether I like it or not. This is where machine learning comes in.

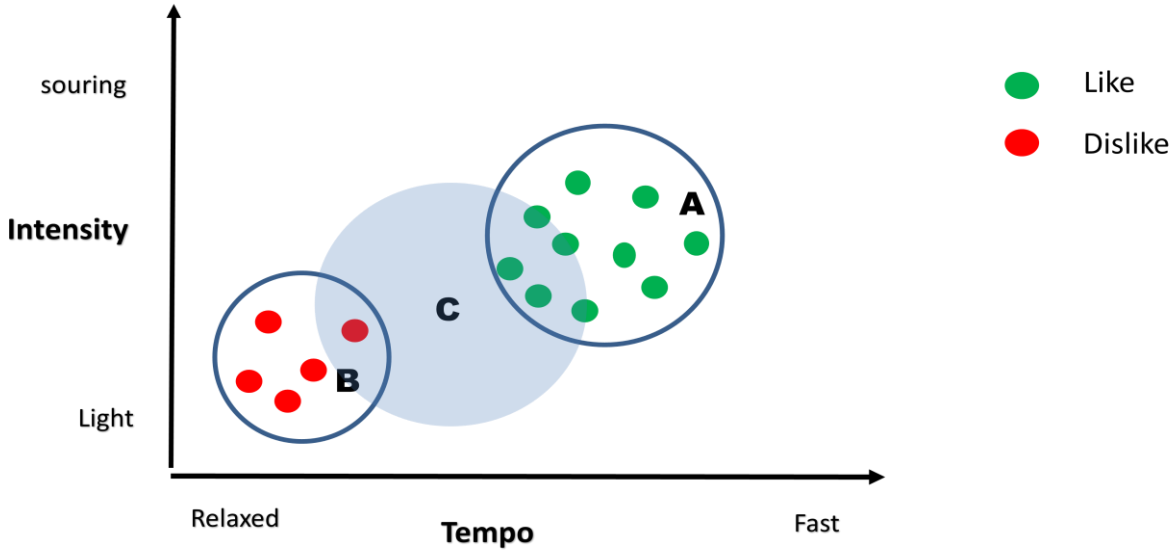


Fig 2.3: Predicting output of new data input.

Now, if I draw a circle around this song C, we see that there are 5 votes for like whereas 1 vote for dislike. Going for majority votes, you can definitely say that I will definitely like it. When choices become complicated, it learns the data, builds the prediction model. And, when new data point comes in, it can easily predict for it. This is basic Machine Learning algorithm called K-nearest neighbors. The data that you provide to a machine learning algorithm can be just inputs or input-output pairs. Supervised learning algorithms require input-output pairs (i.e. they require the output). Unsupervised learning requires only the input data (not the outputs) .

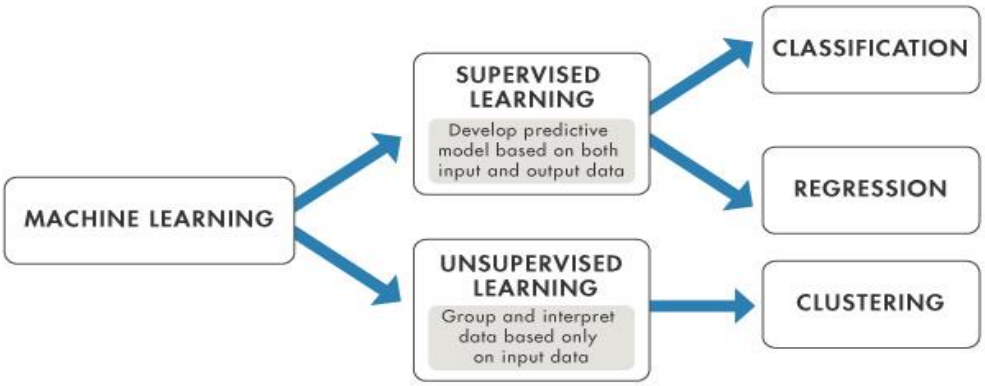


Fig 2.4: Machine learning and its type

2.1.1 Here is how, in general, supervised algorithms work:

You feed it an example input, then the associated output. You repeat the above step many many times. Eventually, the algorithm picks up a pattern between the inputs and outputs. It uses labelled data to train the model. Now, you can feed it a brand new input, and it will predict the output for you .

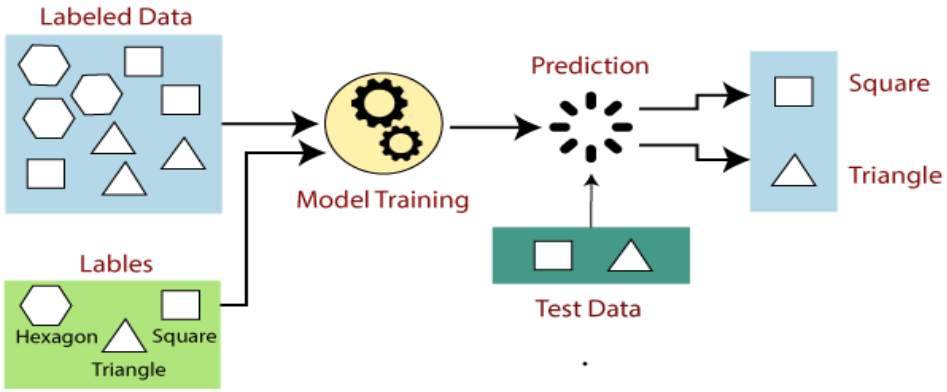


Fig 2.5: Supervised Machine Learning.

2.1.2 Here is how, in general, unsupervised algorithms work:

You feed it an example input (without the associated output). You repeat the above step many times. Eventually, the algorithm clusters your inputs into groups. Now, you can feed it a brand new input, and the algorithm will predict which cluster it belongs with.

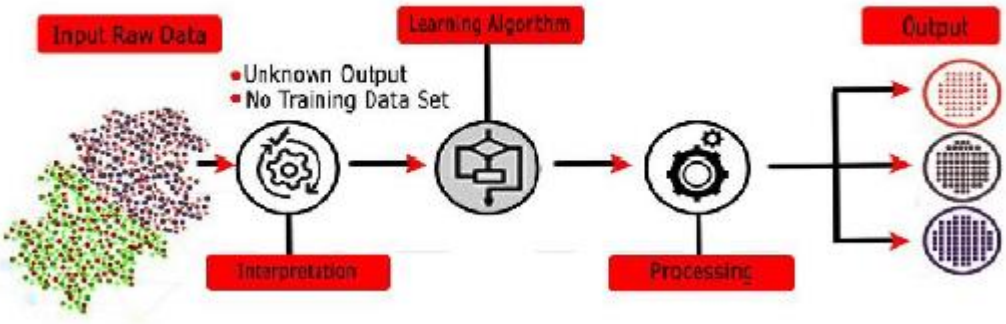


Fig 2.6: Unsupervised Machine Learning.

Supervised learning has to do with the fact that we gave the algorithm a data set in which the "right answer" were given. In regression, the model predicts continuous valued output, whereas in classification, the model predicts discrete valued output. In supervised learning, we are provided with a dataset and already know what our correct output should look like, more formally, this data set is called training set. [9]

2.2 MODEL REPRESENTATION:

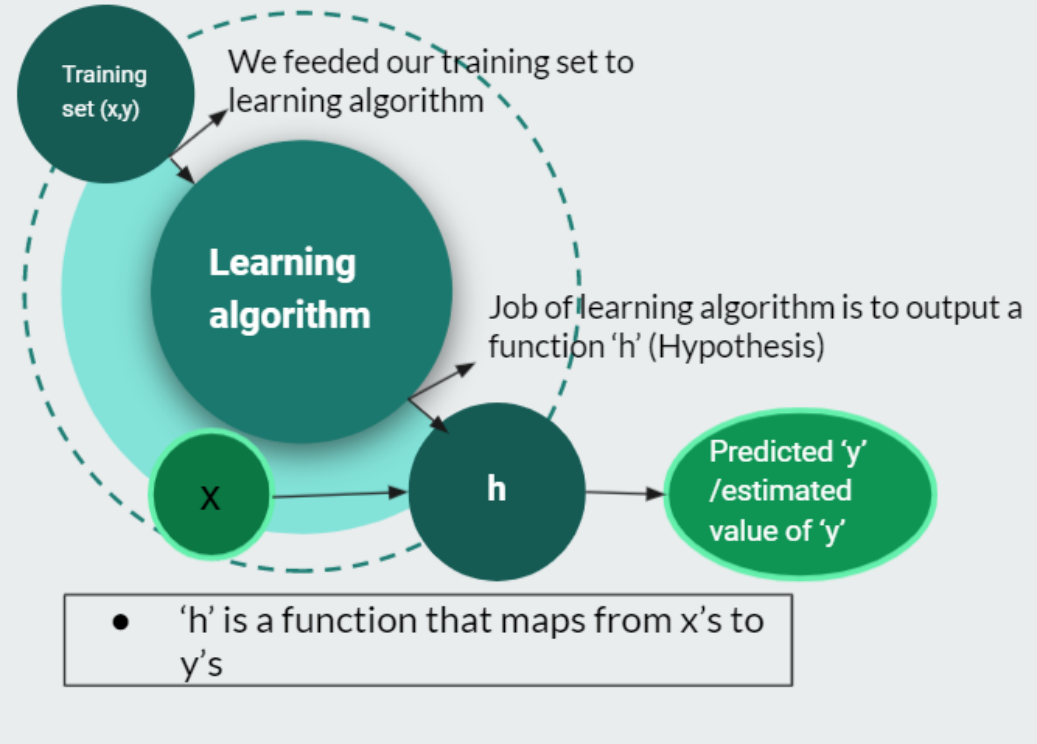


Fig 2.7: Basic algorithm of a machine learning model

X's = "input" variable/features

Y's = "output" variable/ "target" variable

m= number of training examples

We feed our training set (x,y) to our learning algorithm. The job of learning algorithm is then to output a function which by convention is usually denoted by 'h', and 'h' stands for **hypothesis** . The job of hypothesis is a function that takes as input the value of 'x' (feature), and it tries to give the predicted value of 'y' for the corresponding 'x'. So 'h' is a function that maps X's to Y's . A pair $\left(x^{(i)}, y^{(i)}\right)$ is called training example, and the dataset that we'll be using to learn –a list of "m" training examples $\left(x^{(i)}, y^{(i)}\right)$: $i = 1,....m$ -- is called a training set. To describe the supervised learning problem more briefly an correctly, our goal is, to learn a function h: X to Y , given a training set, so that h(x) is a "good" predictor for the corresponding value of y. This function is called hypothesis.

When the target/input variable we are trying to estimate/predict is continuous, then this learning problem is defined as regression problem. When y can input only a small number of discrete values, it will be called as classification problem. When designing the learning algorithm, the next thing we need to decide is how do we represent this hypothesis h. To explain the basic model, we will predict that y is a linear function of x. That is the dataset, and what this function is doing, is predicting that y is some straight line function of x.

$$h_\theta(x) = \theta_0 + \theta_1 x$$

Why a linear function? Well, sometimes we will want to fit more complicated , perhaps non-linear function as well, but since this case which is linear is the basics and simple, we will explain with this example first of fitting linear functions. How to choose θ s? How do we come up with a value of θ_0 and θ_1 that corresponds to a good fit to the data. The idea is, we get to choose our parameters θ_0 and θ_1 so that $h_\theta(x)$ (meaning the value we predict on input x) is at least close to the values y for the examples in our training set. For $h_\theta(x)$ to be close to y, we must minimize the difference between $h_\theta(x)$ and y.

2.3 COST FUNCTION:

$$\underset{\theta_0,\theta_1}{\text{minimize}} \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

This finds us the value of θ_0 and θ_1 so that average over 1/2m times the sum of square errors between my prediction on the training set minus the actual values of the houses on the training set is minimized. That is cost function:-

$$J(\theta_0,\theta_1) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2$$

$\underset{\theta_0,\theta_1}{\text{minimize}}\ J(\theta_0,\theta_1)$ This J is our cost function (or squared error function). Why do we square the error? Squared error function is a reasonable choice and work well for most regression problems. By using a cost function we can calculate the accuracy of our hypothesis function. This takes an average difference of all the results of hypothesis with

inputs from x's and the actual output y's. To break it apart, it is $\frac{1}{2}\bar{x}$ where $\bar{x}$ is the mean of the squares of $h_\theta\left(x^{(i)}\right) - y^{(i)}$, or the difference between the actual value and the predicted value. This function is also called the "mean squared error" or "Squared error function". For the smoothness in calculation of gradient descent the mean is halved (1/2), so that the square function's derivative term (containing a factor of 2) will cancel out the ½ term. [10], [11]. So now, our goal is to minimize cost function J with respect to the parameters .

2.4 GRADIENT DESCENT:

Now we will talk about an algorithm called gradient descent for minimizing the cost function J. It turns out that gradient descent is not just used in linear regression, but in many of general algorithms, wherever we need to minimize a function. It is actually used all over the place in machine learning. Now we are going to assume that we have some function $J(\theta_0, \theta_1)$. Maybe it is the cost function or some other function we want to minimize. And we want to find out an algorithm for minimizing this as a function of $J(\theta_0, \theta_1)$. So we have a hypothesis function and we know a way of computing this function which can fit perfectly into the data. Now we need to predict the parameters in the hypothesis function. That is where gradient descent helps us. Suppose that we graph our hypothesis function based on its parameter θ_0 and θ_1 (actually we are plotting the cost function as a function of the approximated parameter). We are not plotting x and y itself, rather we are plotting cost function coming out from selecting a particular set of parameters and the hypothesis function's parameter range.

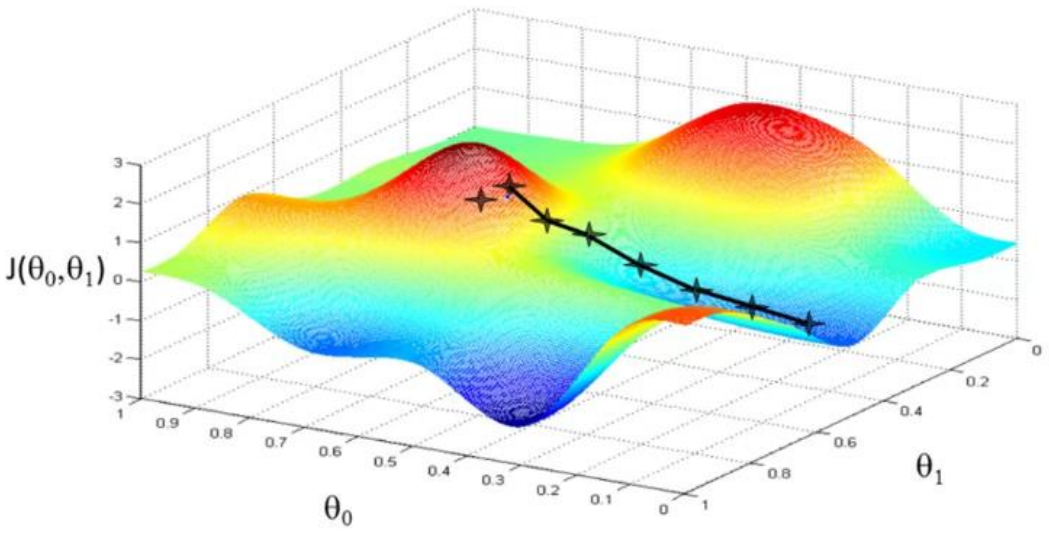


Fig 2.8: Gradient descent.

Now suppose we are trying to minimize our function $J(\theta_0, \theta_1)$. So notice the axes, that are θ_0 and θ_1 on the horizontal axis and J is the vertical axis and so the height of the facet shows J and we want to minimize this J. So, we are going to start off with θ_0 and θ_1 at some point. So imagine picking some value for θ_0, θ_1 and that corresponds to starting at some point on the surface of this function. So whatever value of θ_0, θ_1 gives you some point here. Now suppose this is like a landscape of some grassy park , with two hills like so. And let us imagine that we are physically standing at that point on the hill. In gradient descent , what we are going to do is , we are going to rotate 360 degrees around, and look around us everywhere, and think, if we were to take a little baby steps in a particular direction, and we want to go downhill as quickly as possible, what direction do we take that small steps in? If I want to go down, so I want to physically walk down this hill as quickly as possible. Turns out that if we are standing at that point on the hill as shown in figure 11, we look all around and we find that the best direction is to take a little step downhill in that particular direction as shown. Now we are at a new point on the hill. We are going to do the same thing all over again. And if we do that and take another step, we take a step in steepest descent. And then we keep going. Take another step, another step, and so until we converge to this local minimum. The point on our graph will be the result of cost function using our hypothesis with those specific theta parameters. We will come to know that we have reached our minimum when our cost function is at the bottom of the surface in our graph. We did this by calculating the derivative of our cost function. The slope of the tangent at that point is the derivative and it will tell us a path to move towards. We will move down the $J(\theta_0, \theta_1)$ in the direction with the steepest descent. The size of each of this step is decided by the parameter α, which is known in this case as the learning rate.

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \quad \text{for } (j = 0 \,\&\, j = 1)$$

The distance between each point in the plot represents a step decided by the parameter α. A smaller α will give us smaller step and a large α will give us large steps. The step taken in that particular path is decided by the partial derivative of $J(\theta_0, \theta_1)$. At each iteration j, we should update the parameter $\theta_0, \theta_1, \theta_2, ...\theta_n$ simultaneously. Correct simultaneous update:

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\theta_1 := \text{temp1}$$

Incorrect simultaneous update:

$$\text{temp0} := \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$$
$$\theta_0 := \text{temp0}$$
$$\text{temp1} := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$$
$$\theta_1 := \text{temp1}$$

[12]

2.5 MODEL REPRESENTATION IN NEURAL NETWORKS:

Now we will understand how to represent hypothesis function using neural network. At a very basic level, neurons are computational unit and it takes electrical impulse (called spikes) as a inputs (that is dendrites) which are then directed to outputs (called axons). In our model, input features $x_0, x_1, x_2, ...x_n$ are dendrites, and the hypothesis function gives output as a result. In this neural network model, x_0 input is called bias unit, and it is equal to 1. The functions we use in classification, we will use them here as well. $\frac{1}{1+e^{-\theta^T x}}$, which is also called as sigmoid activation function. Here, our θ parameters are known as weights.

2.5.1 Sigmoid function and ReLU function as activation function:

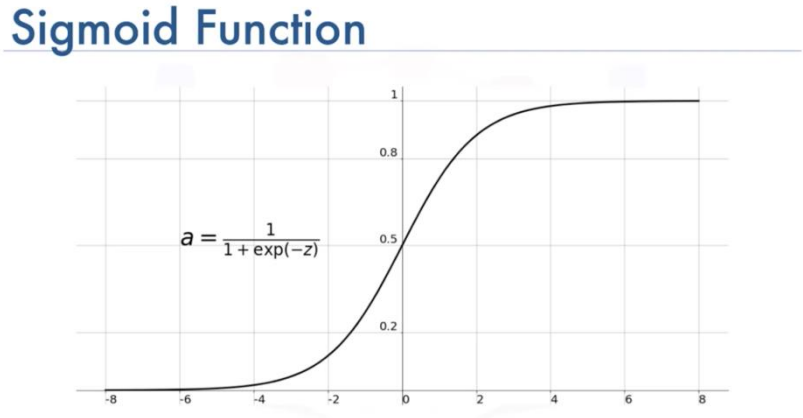


Fig 2.9: sigmoid function

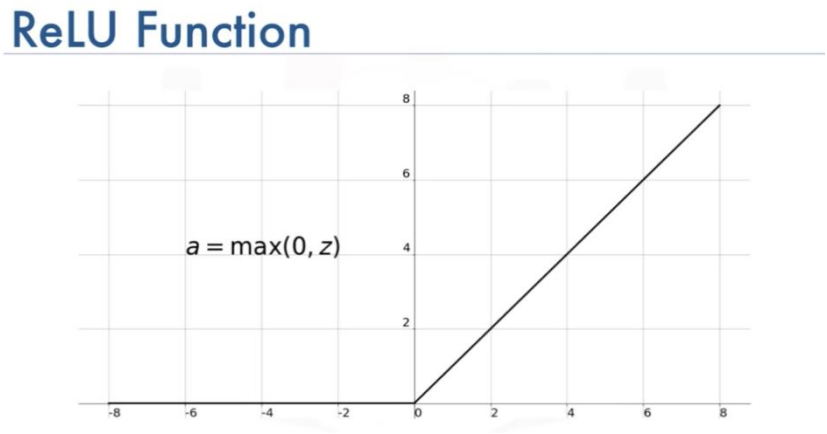


Fig 2.10: ReLU activation function

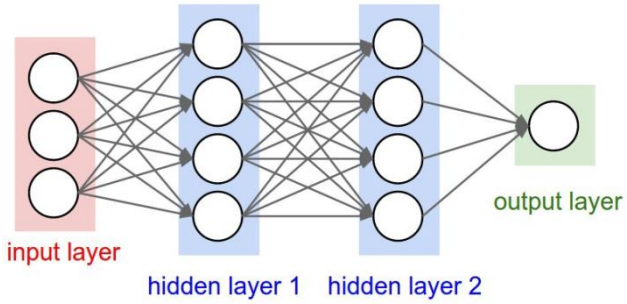


Fig 2.11: Neural network structure

The input nodes which has the value 1 (bias unit) which are also called input layer goes into another layer (another node) which is called hidden layer, and after passing these hidden layers which are laid down according to the problem we have, the output gives us the hypothesis function, which is called output layer. We will label these hidden layer nodes as $a_0^2,....a_n^2$ and call them activation unit. $a_i^{(j)}$ = activation of unit I in layer j. $\Theta^{(j)}$ = matrix of weight which controls the function mapping from layer j to j+1. If we make neural network with one hidden layer:

$$[x_0 x_1 x_2 x_3] \rightarrow [a_0^{(2)} a_1^{(2)} a_3^{(2)}] \rightarrow h_\theta(x)$$

The values for each activation node are calculated as:

$$\begin{aligned} a_1^{(2)} &= g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3) \\ a_2^{(2)} &= g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3) \\ a_3^{(2)} &= g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3) \\ h_\Theta(x) = a_1^{(3)} &= g(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}) \end{aligned}$$

We can see that we have calculated our activation nodes by using matrix of parameters of 3×4 dimension. Each row of the parameter is multiplied with our inputs to obtain the value for one activation node. The main output, that is the hypothesis function is equal to the activation function (g) multiplied to the sum of values of the activation nodes, each of which are multiplied by another parameter matrix $\Theta^{(j)}$ which contains the weights for the second layer of nodes. Each layer get its own matrix weight $\Theta^{(j)}$. Now we will implement something. We will vectorize the above functions. We will define a new variable $z_k^{(j)}$, which encloses the parameter inside our g (activation) function :

$$\begin{aligned} a_1^{(2)} &= g(z_1^{(2)}) \\ a_2^{(2)} &= g(z_2^{(2)}) \\ a_3^{(2)} &= g(z_3^{(2)}) \end{aligned}$$

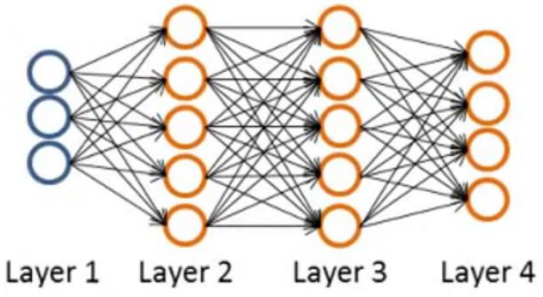
Or, we can say that, for layer j=2 and node k, the variable z will be :

$$z_k^{(j)} = \Theta_{k,0}^{(1)} x_0 + \Theta_{k,1}^{(1)} x_1 + ...\Theta_{k,n}^{(1)} x_n$$

The vector representation is :

$$x = \begin{bmatrix} x_0 \\ x_1 \\ .... \\ x_n \end{bmatrix} \quad z^{(j)} = \begin{bmatrix} z_1^{(j)} \\ z_2^{(j)} \\ .... \\ z_3^{(j)} \end{bmatrix}$$

2.6 FORWARD PROPAGATION ALGORITHM:



$$a^{(1)} = x$$
$$z^{(2)} = \Theta^{(1)} a^{(1)}$$
$$a^{(2)} = g(z^{(2)}) \quad (\text{add } a_0^{(2)})$$
$$z^{(3)} = \Theta^{(2)} a^{(2)}$$
$$a^{(3)} = g(z^{(3)}) \quad (\text{add } a_0^{(3)})$$
$$z^{(4)} = \Theta^{(3)} a^{(3)}$$
$$a^{(4)} = h_\Theta(x) = g(z^{(4)})$$

Fig 2.12: Forward and backward propagation

Intuition: $\delta_j^{(l)} =$ "error" of node j in the layer "l". For each output unit (layer L=4):

$$\delta_j^{(4)} = a_j^{(4)} - y_j$$

The first term (activation of unit j in layer 4) can also be written as:-

$$\left(h_\theta(x) \right)_j$$

This delta term is just the difference between a hypotheses output and what was the value of y in our training set , where y_j is the jth element of the vector value in our labelled training set.

2.7 BACKPROPAGATION ALGORITHM:

For minimizing our cost function, we used gradient descent in linear regression, but when we are dealing with neural networks, we use Backpropagation for minimizing the function. Therefore, our goal is to calculate:

$$\min_{\Theta} J(\Theta)$$

That is, we are minimizing the function J using an optimal set of parameters in Θ . Now we will see the equations we used to calculate the partial derivative of $J(\Theta)$:

$\frac{\partial}{\partial \Theta_{i,j}^{(l)}} J(\Theta)$. Given training set, $\{(x^{(1)}, y^{(1)}) \ldots (x^{(m)}, y^{(m)})\}$

Set $\Delta_{i,j}^{(l)} := 0$ for all (l,i,j) (hence, we end up having matrix full of zeros).

For training example t= 1 to m:

1. Set $a^{(1)} := x^{(t)}$
2. Perform forward propagation to compute $a^{(l)}$ for l=2,3,…L
3. Using , compute $\delta^{(L)} = a^{(l)} - y^{(t)}$

   Where total number of layer is L and $a^{(L)}$ is the outputs of activation unit for the last layer or we can ay hypothesis, and it is a vector. So, simply, the last layer's error values are the difference of true results (output of activation unit) and the correct outputs of y.

4. Compute $\delta^{(L-1)}$, $\delta^{(L-2)}$,… $\delta^{(2)}$, using $\delta^{(l)} = \left(\left(\Theta^{(l)} \right)^T \delta^{(l+1)} \right) .* a^{(l)} .* (1 - a^{(l)})$

   Where $g'(z^{(l)}) = a^{(l)} .* (1 - a^{(l)})$

5. $\Delta_{i,j}^{(l)} := \Delta_{i,j}^{(l)} + a_j^{(l)} \delta_i^{(l+1)}$

   Hence we update our new matrix Δ:

$$D_{i,j}^{(l)} := \frac{1}{m} (\Delta_{i,j}^{(l)} + \lambda \Theta_{i,j}^{(l)}), \text{ if } j \neq 0$$

$D_{i,j}^{(l)} := \frac{1}{m} (\Delta_{i,j}^{(l)})$ if j=0 . Thus we get $\frac{\partial}{\partial \Theta_{i,j}^{(l)}} J(\Theta) = D_{i,j}^{(l)}$

2.8 K- NEAREST NEIGHBOUR (KNN) CLASSIFICATION:

The KNN algorithm is one of the very basic supervised machine learning algorithm which can do both multi-class and binary classification. When we say it is a basic supervised algorithm, we mean that the data we will use for training this KNN model will be a labelled one. In the classification problem, we have to find the discrete values which are possible estimated outcomes. And the thing which model does is that it finds out to which target class does a given data point belongs to. The number of possible target classes is 2 in binary classification problem. Whereas, when there are more than two possible target classes, it is called multi-class classification problem. Since, we know that KNN is a non-parametric algorithm, which means while training a KNN classifier, we don't need to train our model by iterating over the training set which makes our set of parameter optimized, where we apply mathematical equations to minimize the error between the training set and the predicted outcome. Rather, in a KNN classifier model, we will train it by fitting or saving all the training data points in the computer memory, which requires only one training cycle. Now, at the inference stage, when the model will predict the target class for a new data point, the model will normally compare the new data with the data instances which is being saved in the computer memory. And finally, looking at this comparison, the model allocate this new data point to its target. What is this comparison we are exactly talking about? The algorithm name K-nearest neighbour in itself answers this question. In the first step, model computes the distance of the new data from every single data point within the saved training data in memory. The next step is, those training data points are selected which are nearest to the new data on the basis of computed distance. The model select 'k' number of training data. At the end, the model's algorithm compares the target class/label of these 'k' points which are closest neighbor to the new data point. The label which has the maximum frequency within these k-nearest neighbors is allocated as the target class to this new data point. While computing the distance between training data point and new data point, we will use the Euclidean distance formula. [13], [14]

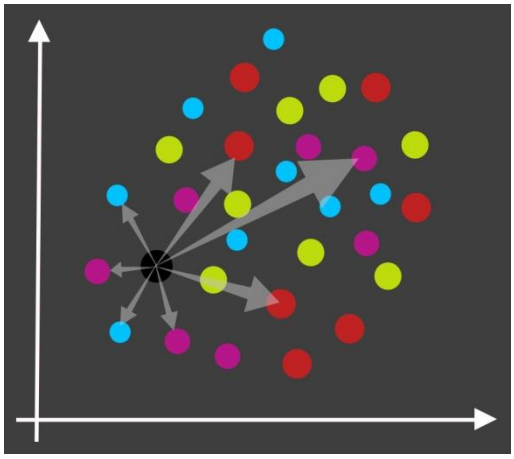


Fig 2.13: Euclidean distance of every training dataset from new

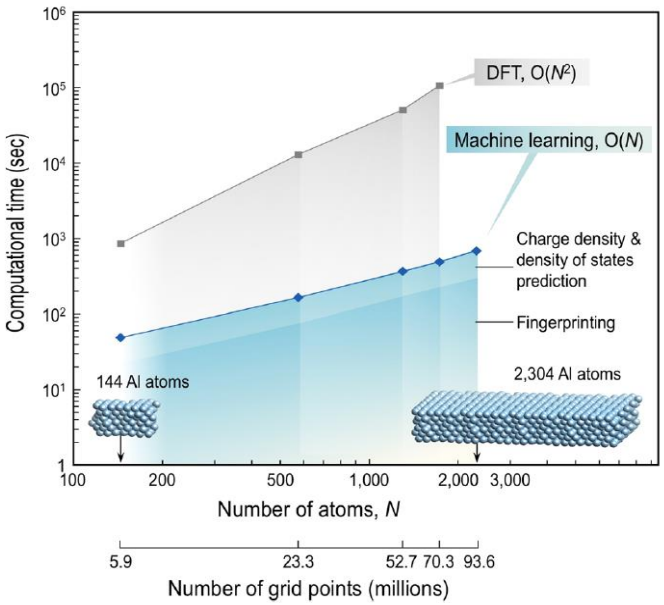
CHAPTER 3. APPLICATION OF MACHINE LEARNING IN DFT:

3.1 Fundamental idea:

The idea is to make a Machine Learning model using neural networks architecture which is trained on previously calculated labelled DFT results at millions of grid point. This prototype gives a copy of Kohn-Sham DFT but of high quality, whose order of magnitude faster than the solution which comes out form K-S DFT. In Machine Learning, we have different models, or neural network architecture which we use to see the pattern within the data. Either we can make use of model which is already being made, to study the data, or we can make our own model, based upon the complexities of the data, and the predicted output (target variable) we need. We can use Machine Learning based model to efficiently understand the function of Kohn-Sham equation quickly, which we discussed in previous chapters. Ultimately, we can make use of big databases of crystal structures for screening the compounds which have the optimal properties using Machine Learning. Now, if we want to predict electronic structure of 2D ultrathin materials accurately and to predict new materials with desired properties, we must decide how to represent the crystal structure as a set of numbers, which we also call fingerprints or descriptor, since the neural network algorithm understands only numbers. These set of numbers are called fingerprints. [15], [16]

3.2 Bottleneck and problems which inspired us to use Machine Learning :

We know that DFT consumes a lot of time since it has a lot of quantum calculation and approximation methods. Every time we solve the Kohn-Sham equation separately, a huge amount of data is being produced. For example, there are millions of grid points (points on the Cartesian co-ordinate) for around 100 of aluminium atom, and when we calculate the charge density of these 100 atoms, the data is produced for each of the grid point. A lot of data gets evaluated but the use of it is not proportionate to the amount of data which is being produced. Although, it gives accurate values for the corresponding input material, it is not suitable for producing result for large database, as it takes a lot of computational time. [15], [16], [17]



.

Fig 3.1: Computational time and scaling of density functional theory (DFT) vs machine learning (ML) for electronic structure predictions. DFT shows near-quadratic scaling, whereas the ML prediction algorithm shows perfect linear-scaling and is orders of magnitude faster than DFT.

3.3 Main idea of Machine Learning model:

The idea is to use the training data set, which contains the features, that is the input, with already know outputs known as target or labels. We can pass the dataset through the model and it optimizes its internal parameter, which are also called weights as discussed in second chapter, which reduces the error to obtain correct results. Once the model gets trained, the computation or calculation of the desired properties is instant. Internal parameters are basically the patterns found inside the training dataset. Many, researchers have created a substitute Machine Learning model to predict DFT's primary and secondary outputs. The goal is to create strict linear ML model which can predict the DFT's primary output much faster than what DFT itself could have calculated. As we increase the number of atoms, the computational time increases quadratically (or squarely), but when we use ML models, the computational time decreases remarkably, and as we increase the number of atoms, the computational time increases linearly. The aim is to bypass the direct solution, and understand the function of Kohn-Sham equation and learn the pattern between inputs and the outputs calculated using Kohn-Sham equation. [19]

Now, there are two ways to predict new materials, without performing trial and error experiment, which can take a huge amount of time. Using crystal database, we can search within existing materials. Or, by the means of computer algorithm, we can create a whole new structures. And we also know that , if we want to use any prediction model, we must specify the desired properties which we need to optimize. And once we specify this, we need to have a method to evaluate them from crystal structure. A very basic and simple approach is to represent the materials in terms of grid point of every atom or their Cartesian coordinate. This technique does not produce good results. Because, if we represent them this way, the material's structure which are actually equivalent might have different fingerprint. A proper descriptor will remain same if we apply rotation, reflection or transmission. Such fingerprint are complex and tough to make when the datasets have many atom type. The descriptors are what actually being feed into the model, which then optimizes this and find pattern within this.

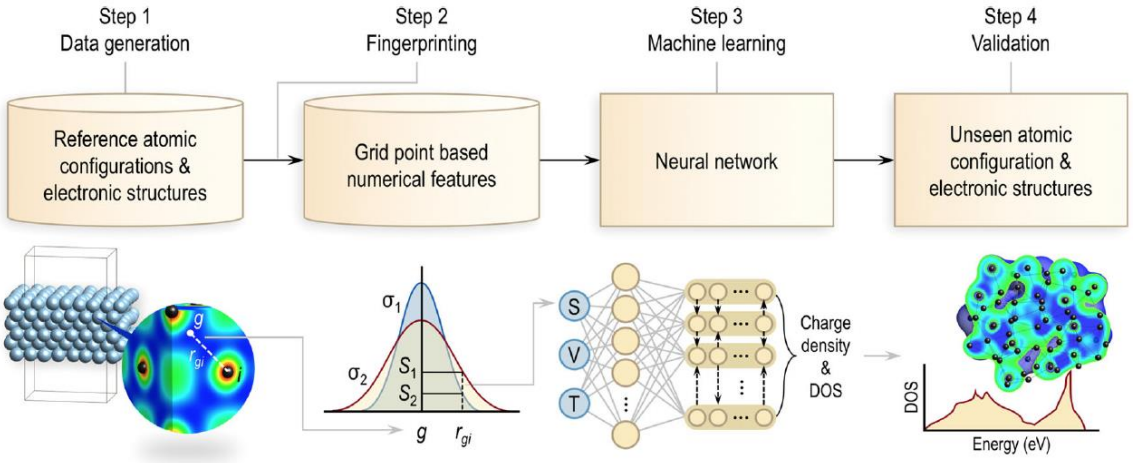


Fig 3.2: Overview of the process used to generate surrogate models for the charge density and density of states.

From the molecular dynamics trajectory, they took some snapshots of aluminium atom. These snapshots displayed abundant varieties of structural environment, which were then used as an input or training set for training the ML model. Now, to map the surroundings around a grid point, fingerprints/descriptors which were taken into consideration were rotationally invariant. The mapping from the input to the labelled output was assimilated using neural network. Once the model or the neural network architecture is trained, we can feed the new dataset and then we can compute their electronic structure. [18]

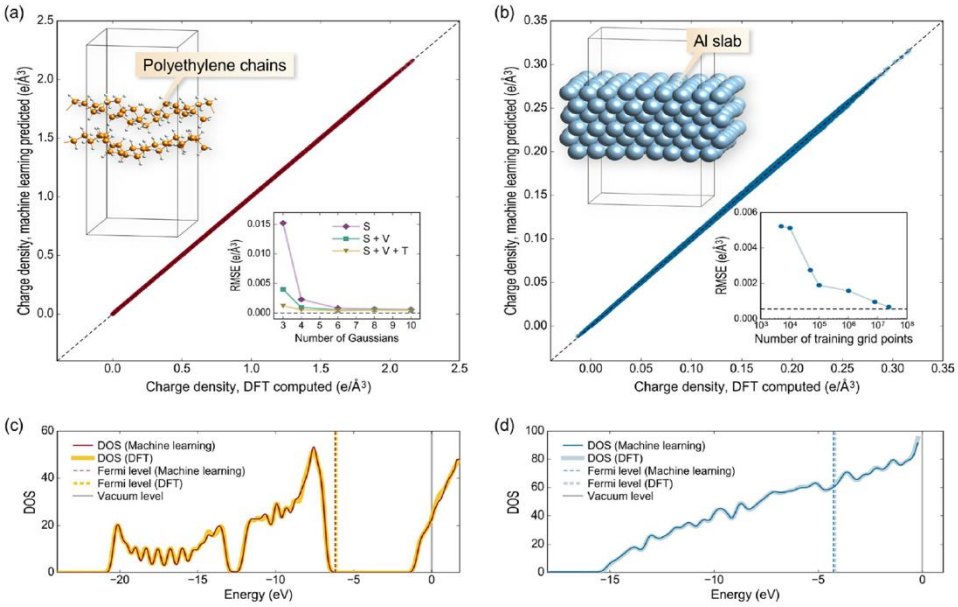


Fig 3.3: Parity plot for the machine learning vs density functional theory (DFT) charge density prediction for the unseen snapshot of a polyethylene (PE) and b aluminum (Al).

The plots of figure 19, shows the structured improvement in the accuracy of the model when we used the vector and tensor descriptor. We can also see that, as we increased the number of Gaussians which were used to sample the local surrounding of each atom, the accuracy increased as well. The plot (b) shows that when we feed more grid points into our training dataset, the error within the model also reduced. [15]

CHAPTER 4. RESULTS AND DISCUSSION:

4.1 Density of states and Band Structures:

We had considered vanadium oxide (V_2O_5) as our material and calculated various properties, to understand how DFT calculations work. The peak represents the available states for the corresponding energy level. On observing the edge of the conduction band, the valence shell electron of Vanadium (d orbital) and Oxygen (p orbital), undergo hybridization. The two bands are separated by some empty space and this means that there are no states available for occupancy within this range hence signifying the Band gap which comes out be 1.9 eV which is nearly equal to the experimental value of 2.2 eV. The indirect band gap for V_2O_5 was observed as 1.7 eV and direct gap as 2.007 eV. The effective mass of electron (m_e*) and holes (m_h*) was calculated 1.823 m_0 and 4.584 m_0 respectively, in terms of rest mass of electron (m_0). The valence band maxima was observed at -1.183 eV and conduction band minima at 0.0564 eV (Figure 4).

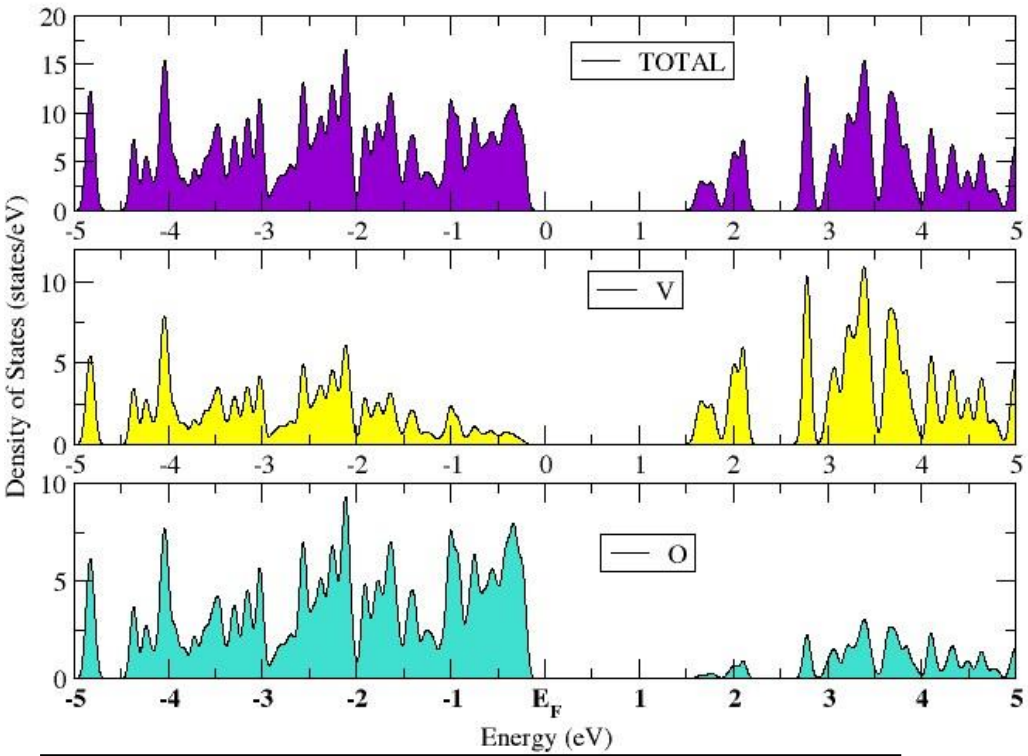


Fig 4.1: Density of states

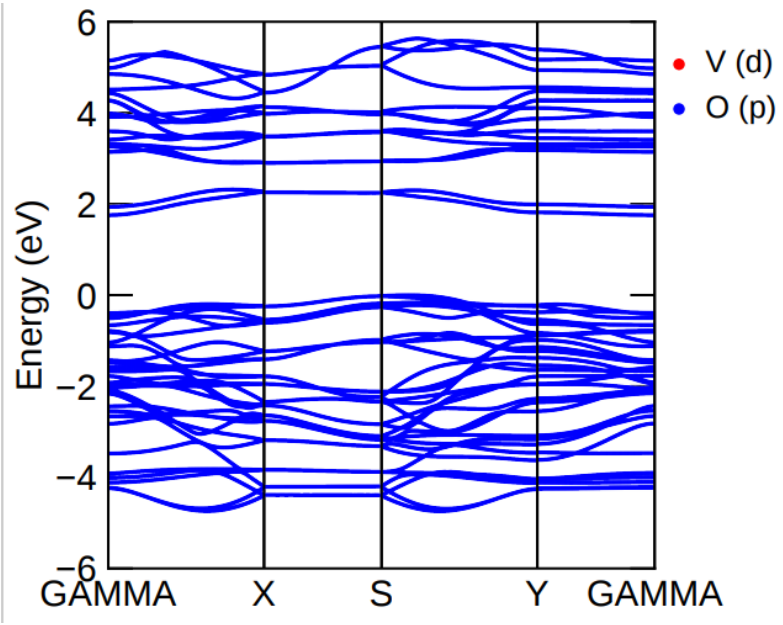


Fig 4.2: Band Structures of V_2O_5

4.1.1 Absorption spectra, Reflectivity and Refractive index

The absorption coefficient when extrapolated from the first peak gives the band gap of the material. The energy ranges from 2-5 eV corresponds to the ultraviolet region of the Solar Spectrum.

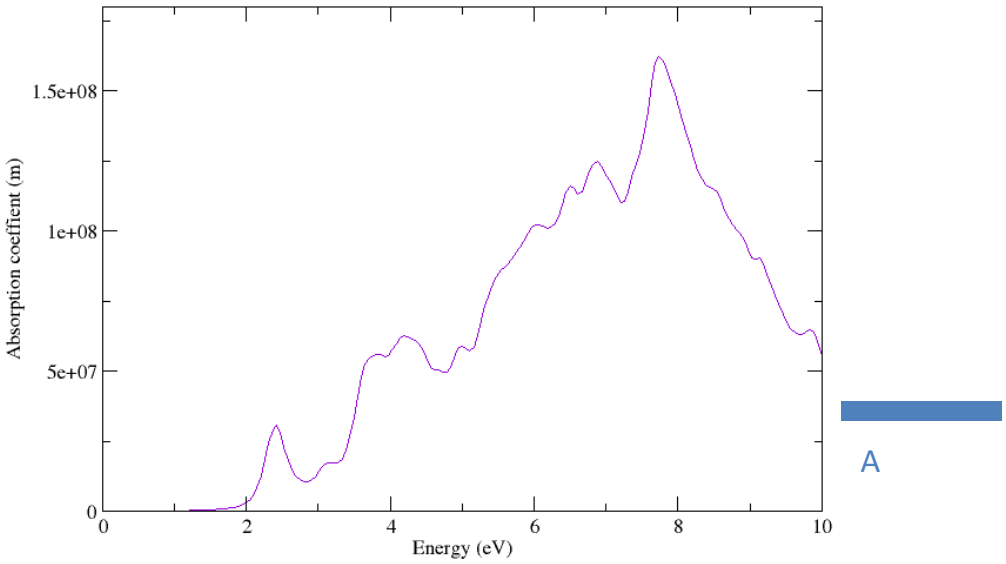
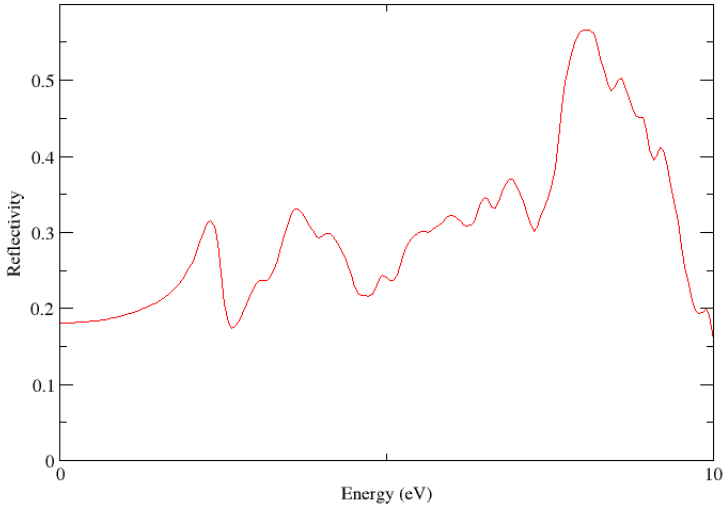


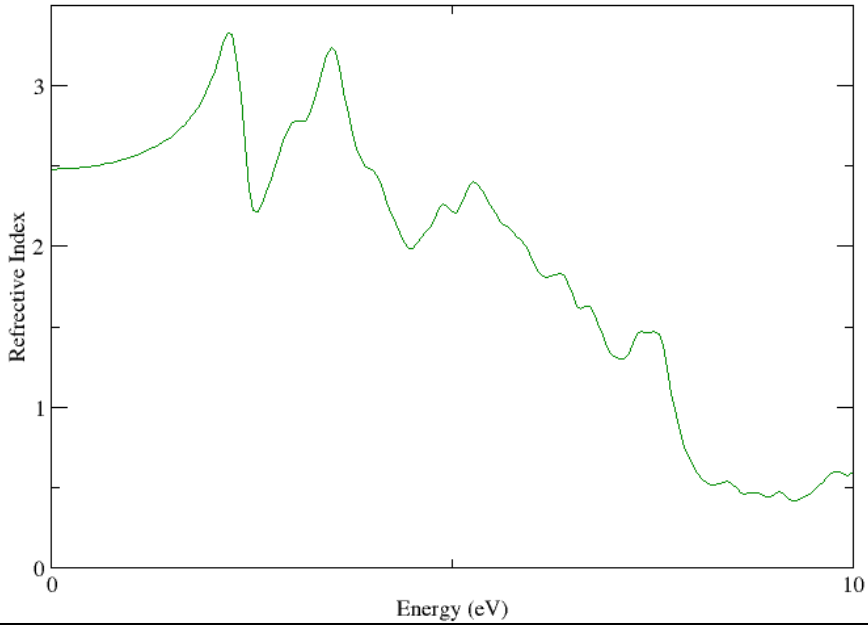
Fig 4.3(A): The Absorption coefficient **α(λ)**

The Absorption coefficient α(λ) describes the intensity attenuation of the light passing through a material. It can be understood as the sum of the absorption cross-sections per unit volume of a material for an optical process. The higher α(λ), the shorter length the light can penetrate into a material before it is absorbed.



B

Fig 4.3(B): Reflectivity is an optical property of material (in our case, it is V_2O_5), which describes how much light is reflected from the material in relation to an amount of light incident on the material.



C

Fig 4.3(C): it can be seen that Refractive index gets negative as the energy increases (or, we go beyond visible range).

This is because widely used and a well-established thin-film technique called oblique angle deposition can be used to deposit thin films that refract visible light negatively, and this negative refraction can occur over a broad range of wavelength.

THEORY:

$$(\tilde{n}+i\,\tilde{k})^2 = \varepsilon_1+i\,\varepsilon_2$$

$$\tilde{n}^{\,2} = ((\varepsilon_1^2+\varepsilon_2^2)^{1/2}+\varepsilon_1)/2$$

$$\tilde{k}^2 = ((\varepsilon_1^2+\varepsilon_2^2)^{1/2}-\varepsilon 1)/2$$

$$\alpha_{abs} = 2*E*\tilde{k}/(\hbar*c)$$

$$R = ((\tilde{n}-1)^2+\tilde{k}^2)/((\tilde{n}+1)^2+\tilde{k}^2)$$

$$L = \varepsilon_2/(\varepsilon_1^2+\varepsilon_2^2)$$

where, $\varepsilon_1+i\,\varepsilon_2$ is complex dielectric function, $\tilde{n}$ is refractive index, $\tilde{k}$ is Extinction coefficient, α_{abs} is absorption coefficient, R is reflectivity, $\hbar$*=6.58211951E-16eVs, c=2.99792E8 m/s

4.2 Energy Band Gap screening :

In our project, we started with calculating different properties of V_2O_5 to get our hands on DFT calculations, and after that, we made a Machine learning model, which screened those 2D materials, which had a band gap between 1.2 to 1.3. Once the model got trained, it was capable of classifying the new datasets, that is, the new 2D materials. The advantage we got here due to our Machine Learning model is that, we were able to sort those materials whose

band gaps lied between 1.2 to 1.3 eV very easily, and the model calculated the band gaps of material without using any formula.

To train our model, we took inputs (features) and the labelled output from the material database. The problem of converting the input datasets (that is the 2D materials) into the set of numbers called fingerprints was tackled in a unique way. We used already calculated results of materials from the material database called C2DB (computational 2D database). Instead of converting the materials in some set of numbers, we calculated their wavelengths using their energy band gaps, and instead of giving materials as input, we made their wavelengths as our input datasets. After that, we trained our model over the input datasets (which was wavelengths of materials) and the corresponding outputs (that was the band gaps), using K-nearest neighbour algorithm from SKLearn library in python. After the model was trained, we gave new datasets as input, that is, **we gave wavelengths of new material (whose band gaps were not classified) as inputs**, which were not involved in training the model, and our model classified these new wavelength inputs into two different classes. One is where the materials have a band gap between 1.2 to 1.3 eV and the other class is of those materials whose band gap doesn't lie in this range. So, the output is telling us which input material's band gap lies between 1.2-1.3 eV and which input material lies outside of it. Ultimately, we were able to sort our materials, without using any pre-defined formula, and just on the basis of the data we entered. The script we used and the scatter plot are shown below.

4.2.1 Importing the data

```
In [1]: import pandas as pd
        import numpy as np

In [2]: # Importing the Data
        data = pd.read_csv(r'....Energy_band_gaps_n.csv',names = ['Formula','Band_gap','Crystal_type'],header=0)

In [3]: # Part of the Data imported
        data.head()
```

Out[3]:

	Formula	Band_gap	Crystal_type
0	Ca4As4	0.998	AB-14-e
1	Mn2Se2	0.000	AB-129-bc
2	O8Te4	2.667	AB2-2-i
3	Ru2F8	0.628	AB4-14-ae
4	V2F8	0.597	AB4-14-ce

```
In [4]: # checking if any null values in data
        sum(data.isnull().values)
```

Out[4]: array([0, 0, 0])

Fig 4.4: Importing the data

Here, we imported the data from our file named Energy_band_gaps_n.csv to our notebook, where we were doing all the coding. In the 3rd part, we imported a part of the data using data.head() funciton, to see how is it arranged, and how many columns do we have and what columns we require. And then in the 4th part, we run a code to check whether there is any null value somewhere in any row or ay column. And the output array contains nothing, which means there is no null value in our data.

4.2.2 Calculation of the λ using band gap from database

Converting the Bandwidth Values to Wavelenth Values by formula

lambda = (h*c)/B.W.

```
In [5]: lambdas = np.array([0.00000124/i for i in data['Band_gap'].values])

C:\Users\HP\Anaconda3\lib\site-packages\ipykernel_launcher.py:1: RuntimeWarning: divide by zero encountered in double_scalars
  """Entry point for launching an IPython kernel.
```

```
In [6]: def get_y_tags():
            y_BW = []
            for i in data['Band_gap'].values:

            # input: Nothing      Output: Array with tag values list(int,int,int...)
            #  Converting BW between [1.2,1.3] to tag 1 and rest to tag 0

                if 1.200 <= i <= 1.300:
                    y_BW.append(1)
                else:
                    y_BW.append(0)
            return y_BW
```

```
In [7]: y_BW = get_y_tags()
```

Fig 4.5: Converting the Band gap value to wavelength values using formula.

In this code, we have taken the band gap values from the material database file, and we have calculated the corresponding lambdas. After doing that, we have classified that the band gap values which will lie between 1.2-1.3 eV will return value 1 and the band gap values which lie outside this range, will return the value 0.

4.2.3 Pre-processing or data wrangling:

```
In [8]: # Converting y tags into numpy array for future processing and removing the lambdas which are having infinite values
        # cause they are outliers

        y_BW = np.array(y_BW)
        y_BW = y_BW[np.where(np.isfinite(lambdas))]

In [9]: # Converting lambda values into numpy array for future processing and removing the lambdas which are having infinite values
        # cause they are outliers
        lambdas = lambdas[np.where(np.isfinite(lambdas))[0]]
        lambdas = lambdas.reshape(-1, 1)
```

Checking number of records on

input: 'lambdas'
and
output: y_BW

which should be same

```
In [10]: len(y_BW),len(lambdas)
Out[10]: (832, 832)
```

Fig 4.6: Removing all those datasets which are outliers in our data

In this code, we have done something which is called data wrangling ore pre-processing, in which, we analyze the data, and see if it requires any kind of sorting, cleaning or scaling so that it can become ready to train the model without hustle and instability. The first thing which we did here, while pre-processing, is that we checked if there are any null values inside our data. Because if there is any value of band gap which is zero, then the corresponding wavelength will come out to be infinite, and we are to give wavelengths as our input. So these data point will become outliers in our dataset. Outliers are those extreme instances inside the data which does not follow the general trend within the data and, can confuse our algorithm. So, we removed all the band gap data which were zero. After doing that, we checked whether the number of inputs (which are lambdas) and the output (which are band gaps) are equal or not. If not, then we have done something wrong while pre-processing, but if they are equal, then we have done the pre-processing of our data correctly. So, at the end of the code, we can see that the number of outputs and inputs which will be given as training set, are equal.

Using Smote Oversampeling to make the Data of both the classes Balanced so that there is no Bias

before oversampelng: class distribution was 805 and 27

After over sampeling: class distribution is 805 and 805

In [11]:
```
def counter(y):

    return '0 :' + str(len([i for i in y if i == 0 ])) +', 1 : ' + str(len([i for i in y if i == 1 ]))

import imblearn
from imblearn.over_sampling import SMOTE

smote = SMOTE()

# fit predictor and target variable
x_smote, y_smote = smote.fit_resample(lambdas, y_BW)

print('Original dataset shape', counter(y_BW))
print('Resample dataset shape', counter(y_smote))
```

```
Original dataset shape 0 :805, 1 : 27
Resample dataset shape 0 :805, 1 : 805
```

Fig 4.7: Smote oversampling is used to make the data of both the classes Balanced so that there is no bias.

When we train our model, there are some problems we encounter, and those are bias and variance of our model. Sometimes, when we are doing classified supervised learning, and have 2 or more classes, the model gets over trained or under trained if the inputs are not balanced. That is, for some classes, the model gets many datasets, and for the rest of the classes, it gets comparatively few datasets, so the model gets biased towards the class which had more datasets, and when we try to predict a class for new input, it classifies it into the class which was overtrained, and predicts incorrectly. So before training the model, we must balance our training set, and we must make sure that the model gets equal number of dataset for each class to train. So, the difference between the actual output and the predicted output shows us the bias of the model. If the bias is high, then the model pays less attention to the training data and oversimplifies the model, which leads to large error in training and test data.

Whereas, variance is actually the variability of predicted output for a given data input or a value which shows us the spread of predicted data. If we have less amount of data, and we want to build an accurate model, then we are trying to build a linear model for non-linear data. Due to this, underfitting happens and the model is not able to capture the hidden pattern inside the data. Such models have high bias and low variance. Similarly, when we train our model over a lot of data, even those extreme outlier datasets also, then overfitting happens and our model catches the noise along with the hidden pattern of the data. So, we must have low bias and low variance to make an accurate model. And for that, we use smote oversampling.

4.2.4 SMOTE Oversampling:

As we have discussed that, imbalance creates bias where the model tries to predict the class with majority datasets. And therefore, we have used a technique to overcome this imbalance problem called undersampling and oversampling. Undersampling decreases the amount of the majority class until the number of inputs is equal to that of the minority class. Whereas, oversampling resample the minority class and majority class.

We used SMOTE technique to oversample our datasets. SMOTE or Synthetic Minority Oversampling Technique is a technique of oversampling. In a normal or basic oversampling, we create duplicate minority data from minority class population. This increases the number of datasets, which have no new information or variance, and is not able to provide variety to the model. SMOTE works by using k-nearest neighbour algorithm to create synthetic data. First, it selects random data from minority class, then k-nearest neighbours from the data are selected. Now the synthetic data is created between random selected data and their random k-nearest neighbours. This process is repeated until the minority class has the same population as that of majority class. So, in the code in figure 26, the class distribution was 805 and 27. If we were to feed this as our training and test data, then our model would have definitely been biased. So we used SMOTE oversampling, and created synthetic data After the oversampling , the class distribution became 805 and 805.

4.2.5 K neighbours classifier and SKLearn library:

Splitting the Data into Train and Test Set

In [12]:
```
from sklearn.model_selection import train_test_split

x_train,x_val,y_train,y_val = train_test_split(x_smote, y_smote, stratify = y_smote, random_state = 10)
```

We are Using KNeighboursClassifier from SKLearn to classify our Materials

Input: array of lambda value (wave length)

Output: array of predicted tags (If B.W. between [1.2,1.3] tag is 1 and for rest tag is 0)

In [13]:
```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=2)

# Train the model using the training sets
model.fit(x_train,y_train)
```

Out[13]:
```
KNeighborsClassifier(n_neighbors=2)
```

Fig 4.8: Splitting the data into train and test and, and applying the model

In this code, we used SKLearn library to split our data into test data and train data. Our train data will be given to the KNeighbourclassifier, or we can say, to our model, to get trained, and once that is done, we will test our model using test data which will act as a new data for our model. Now, to classify, we used KNeighbourClassifier from SKLearn library, and we gave the tag equals to 1 to those materials whose band gap lies between 1.2 and 1.3 and the rest of the materials are given tag equals to 0. The principle of this model is: After the splitting of training data and test data is done, the training data is saved inside the memory. After this, it calculates the distance which is an iterative process, where it calculated the Euclidean distance between the data input in the test dataset and the data points inside the training dataset. After this, the model optimizes and chooses a particular k-value which decides the number of training data points which will be considered while choosing the class of the test data point. Very low k-value makes the model sensitive to the outliers, and very high k-value makes the model stable. After this is done, we need to check how our model has performed. For that, we plotted the data.

4.2.6 Scatter plot of materials with particular band gap:

Plotting our predicted values with x axis as index and y axis as B.W. values

Color Coading is done according to prediction

Green are the elements with B.W. between 1.2 to 1.3

Red are the elements with B.W. between 1.2 to 1.3

In [17]:
```
import matplotlib.pyplot as plt
fig, ax = plt.subplots(figsize=(10, 6))
ax.scatter(range(len(y_val)), [0.00000124/x[0] for x in x_val],c= ['red' if l == 0 else 'green' for l in model.predict(x_val)])
plt.ylim(min([0.00000124/x[0] for x in x_val]), max([0.00000124/x[0] for x in x_val]))
plt.xlabel("Test Sample Index")
plt.ylabel("Band Width")
plt.show()
```

Fig 4.9: Code for scatter plot.

We gave Green colour to those predicted elements whose Band gap was between 1.2 to 1.3 and the rest of the elements were given Red colour. And then we plotted these test data, or elements, and saw that, the predicted values were matching with the actual values, which is a remarkable thing. This shows us that the model is absolutely accurate. We plotted our predicted value, with x axis as index, and y axis as the band gap values.

The plot in figure 29 shows a green line which lies between 1 eV and 2 eV. The dots are actual output of the corresponding input wavelengths. The green colour shows that these outputs are accurately predicted or classified. Because we gave green colour to the class 1 which was band gap with 1.2-1.3, therefore all the output coming inside this range have been classified as green, where as all the elements outside this range were classified with red colour, and therefore all the dots outside this particular range are red.

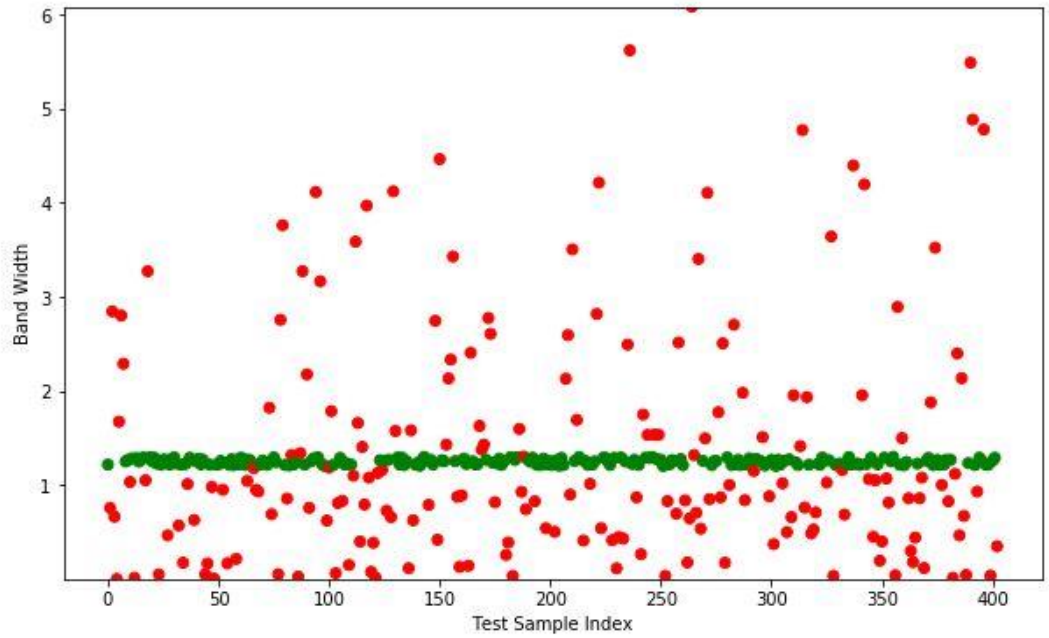


Fig 4.10: Scatter plot of 2D materials screened on the basis of band gap.

In the figure 30, we have zoomed the plot, and looking closely between the range 1.2 and 1.3 eV, there are some red dots as well, which tells us that those particular materials lies between 1.2 -1.3 eV but are predicted or classified as materials which lies outside of this range (that is why they are red). So our model is not perfect, and have a very few amount of materials which are classified incorrectly, but these incorrect predictions are the ones which lies near the class 1.

In [15]:
```
fig, ax = plt.subplots(figsize=(10, 6))
ax.scatter(range(len(y_val)), [0.00000124/x[0] for x in x_val],c= ['red' if l == 0 else 'green' for l in model.predict(x_val)])
plt.ylim(1, 2)
plt.xlabel("Test Sample Index")
plt.ylabel("Band Width")
plt.show()
```

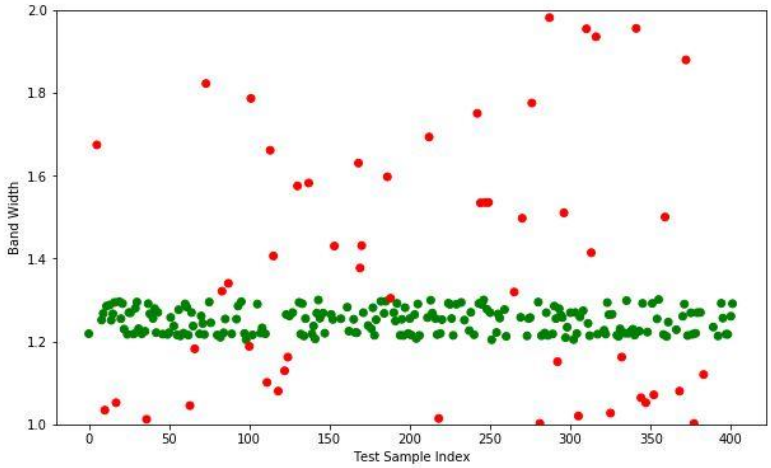


Fig 4.11: Zoomed scatter plot, which shows some incorrect predictions.

So we calculated the F1 score.

Calculating F1 score to calculate our accuracy of model

In [16]:
```
#model.predict(x_val) gives prediction values of our test data by inputing our inputs x_val into model trained
#y_val are actual values of our test data

from sklearn.metrics import f1_score
print('f1 Score is : ' + str(f1_score(model.predict(x_val), y_val, average='micro')))
```

```
f1 Score is : 0.9975186104218362
```

F1 score came out to be 0.9975186, which shows that the model is very accurate, and the error is really small.

CHAPTER 5. CONCLUSION SCOPE FOR FUTURE WORK.

In this project, we have studied Density Functional Theory Machine Learning, and ways to apply the Machine Learning into our DFT, thoroughly. We have done DFT calculations of two materials to get our hands on DFT simulations as well, and we have made initial progress in making a basic machine learning model, which is screening out 2D materials into desired band gaps. Now our next step would be to calculate different properties of different ultrathin materials using Density Functional Theory simulations, so that we can make our own database of inputs and labelled outputs for training the machine learning model, which can replace the function of Kohn-Sham equations mainly used in our DFT computation. Once this is done, we will get our hands on deep learning, and neural networks and which will be the basic architecture of our model, to replace the Kohn-Sham equations function. While solving the problem using neural network, we must know a way to convert our input into valid set of numbers, so we will study different descriptors as well, or we might create our own, to represent our crystal structure Now that we have seen how our simple machine learning model can make things easier for us, we can slowly and consciously apply more complex models into our more complex datasets. And can compute the properties of different crystal structure with large databases.

6. REFRENCES

[1]. Molecular structure and the born—Oppenheimer approximation, R.G.Woolley, B.T.Sutcliffe, Chemical physical letters, https://doi.org/10.1016/0009-2614(77)80298-4

[2]. Slater, J. C. (1970). *Note on the space part of anti-symmetric wave functions in the many-electron problem. International Journal of Quantum Chemistry, 4(6), 561–570.* doi:10.1002/qua.560040603

[3]. Loos, P.-F., & Gill, P. M. W. (2016). *The uniform electron gas. Wiley Interdisciplinary Reviews: Computational Molecular Science, 6(4), 410–429.* doi:10.1002/wcms.1257

[4]. Elber, R., & Karplus, M. (1990). *Enhanced sampling in molecular dynamics: use of the time-dependent Hartree approximation for a simulation of carbon monoxide diffusion through myoglobin. Journal of the American Chemical Society, 112(25), 9161–9175.* doi:10.1021/ja00181a020

[5].https://en.wikibooks.org/wiki/Density_functional_theory/Hartree%E2%80%93Fock_method

[6]. https://www.sciencedirect.com/topics/engineering/hohenberg-kohn-theorem

[7]. https://www.slideserve.com/Sophia/density-functional-theory-the-basis-of-most-modern-calculations

[8]. Pople, J. A., Gill, P. M. W., & Johnson, B. G. (1992). *Kohn—Sham density-functional theory within a finite basis set. Chemical Physics Letters, 199(6), 557–560.* doi:10.1016/0009-2614(92)85009-y

[9]. https://towardsdatascience.com/a-complete-machine-learning-walk-through-in-python-part-one-c62152f39420

[10]. https://towardsdatascience.com/coding-deep-learning-for-beginners-linear-regression-part-2-cost-function-49545303d29f

[11]. Rojc, M., & Mlakar, I. (2020). *A New Fuzzy Unit Selection Cost Function Optimized by Relaxed Gradient Descent Algorithm. Expert Systems with Applications, 113552.* doi:10.1016/j.eswa.2020.113552

[12]. An overview of gradient descent optimization algorithms, Sebastian Ruder, https://arxiv.org/pdf/1609.04747.pdf

[13]. Neural Network Learning: Theoretical Foundations

By Martin Anthony, Peter L. Bartlett

[14]. An Introduction to Statistical Learning: with Applications in R

By Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

[15]. A. Chandrasekharan *et al.* Solving the electronic structure problem with machine learning, npj Comp. Mater., 22, 1 (2019)

[16]. E. Mazhnik *et al.* Application of machine learning methods for predicting new super hard materials, J. App. Phys. 128, 075102 (2020)

[17]: G. Schleder *et al.* From DFT to machine learning: recent approaches to materials science–a review, J. Phys. Materials, 2, 3 (2019)

[18]. Li, L., Baker, T. E., White, S. R., & Burke, K. (2016). *Pure density functional for strong correlation and the thermodynamic limit from machine learning. Physical Review B, 94(24).* doi:10.1103/physrevb.94.245129

[19]. Vu, K., Snyder, J. C., Li, L., Rupp, M., Chen, B. F., Khelif, T., … Burke, K. (2015). *Understanding kernel ridge regression: Common behaviors from simple functions to density functionals. International Journal of Quantum Chemistry, 115(16), 1115–1128.* doi:10.1002/qua.24939