

Mid-term checkpoint SOC 2025

Name-Vihaan Vajifdar

Roll no- 24B2164

SOC Project- Introduction to deep learning

1. Basics of python-

In this I did basics of python which were pretty similar to CS101 course that I already had in the spring semester. So more focus was on learning important python libraries such as Numpy, Pandas and Matplotlib.

Some important syntax I learned related to above libraries are-

NumPy (Numerical Computing)

Array Creation

```
python Copy Edit  
  
import numpy as np  
  
a = np.array([1, 2, 3])           # 1D array  
b = np.zeros((2, 3))             # 2x3 array of zeros  
c = np.ones((3, 3))              # 3x3 array of ones  
d = np.arange(0, 10, 2)          # [0, 2, 4, 6, 8]  
e = np.linspace(0, 1, 5)         # 5 values between 0 and 1
```

Indexing & Slicing

```
python Copy Edit  
  
a[0], a[1:3], a[-1]              # Access elements  
a[a > 2]                         # Boolean indexing
```

Array Operations

python

Copy Edit

```
a + b, a - b, a * b, a / b      # Element-wise operations
np.dot(a, b)                   # Dot product
a.shape, a.reshape(3, 1)      # Shape and reshaping
```

Statistics

python

Copy Edit

```
np.mean(a), np.std(a), np.sum(a) # Mean, std deviation, sum
np.max(a), np.min(a), np.argmax(a) # Max, min, index of min
```

Pandas (Data Analysis)

Creating Structures

python

Copy Edit

```
import pandas as pd

s = pd.Series([1, 2, 3])          # Series
df = pd.DataFrame({'A': [1, 2], 'B': [3, 4]}) # DataFrame
```

Reading/Writing Data

python

Copy Edit

```
df = pd.read_csv('file.csv')      # Read CSV
df.to_csv('output.csv')           # Write CSV
```



Data Inspection

python

Copy Edit

```
df.head(), df.tail()             # First/last rows
df.info(), df.describe()          # Summary & stats
df.columns, df.index              # Column names, index
```

Selection & Filtering

python

Copy Edit

```
df['A'], df[['A', 'B']]           # Select columns
df.loc[0], df.iloc[1]             # Row access by label/index
df[df['A'] > 1]                   # Conditional filtering
```

Modifying Data


python

 Copy  Edit

```
df['C'] = df['A'] + df['B']           # Add new column
df.drop('C', axis=1)                  # Drop column
df.fillna(0), df.dropna()             # Handle missing data
```

Grouping & Aggregation



python

 Copy  Edit

```
df.groupby('A').mean()                # Grouping
df['A'].value_counts()                 # Frequency
```

Matplotlib (Visualization)

python



 Copy  Edit

```
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10, 100)
y = np.sin(x)
```

Line Plot

python

 Copy  Edit

```
plt.plot(x, y)
plt.title("Sine Wave")
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.grid(True)
```





2. Machine Learning and neural networks-

For learning basics of machine learning, I mainly focussed on linear regression and logistic regression.

Labelled data- data that comes with input-output pairs. Each data point includes the features(input) and labels(correct answers)

Unlabelled data- Data that has only the input features, but no output or label.

Supervised learning- The model learns from labelled data, where each input comes with a correct output. The goal is to predict output for new, unseen inputs. Examples include predicting house prices based on features like size, location.

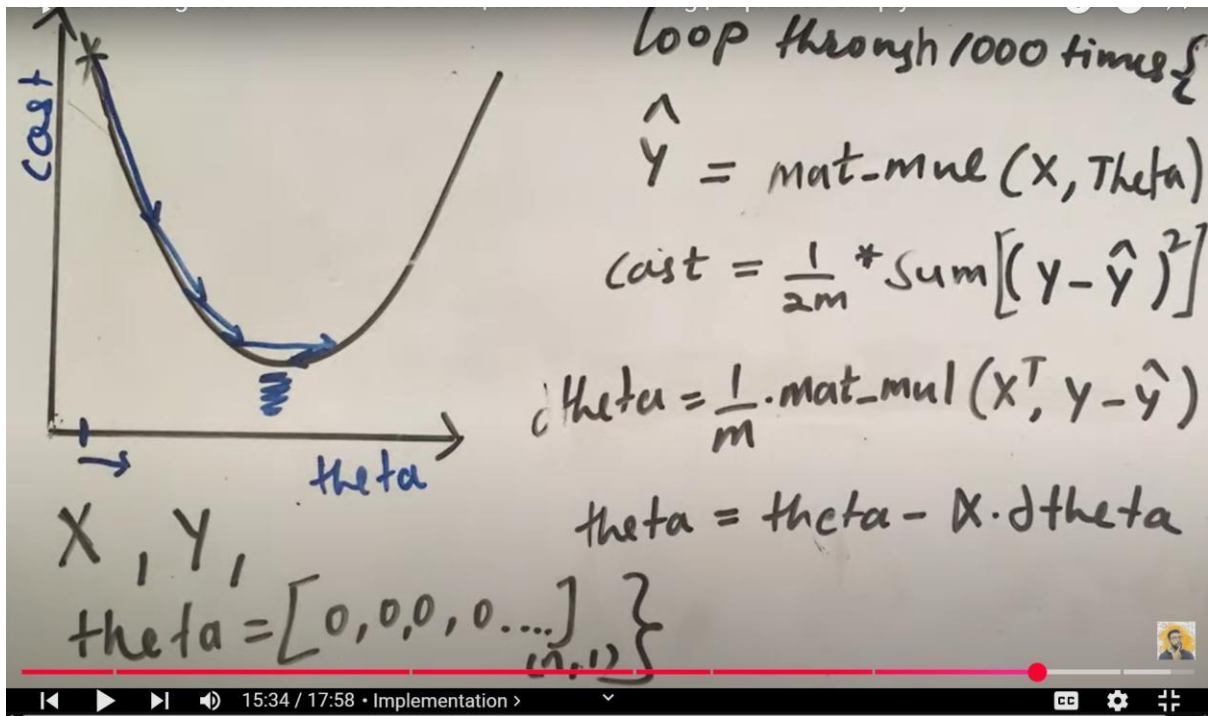
Unsupervised learning- The model learns from unlabelled data, finding patterns or structures without predefined outputs. Examples include grouping customers based on shopping behaviour.

Linear Regression (for prediction of continuous values)

Purpose: Linear Regression is used when the **target variable is continuous** — like predicting house prices, temperatures, or salary.

How it works: It assumes a **linear relationship** between the input variables x_1, x_2, \dots, x_n and the output y . The model learns the best-fitting line (or hyperplane in multiple dimensions) by minimizing the **Mean Squared Error (MSE)** between predicted values \hat{y} and actual values y .

Loss Function (Mean Squared Error):



$$\text{MSE} = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$$

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^m (y_i - \hat{y}_i)^2$$

Logistic Regression (for classification)

Purpose: Logistic Regression is used when the **target variable is categorical**, especially **binary classification** problems (e.g., spam or not spam, pass or fail, disease or no disease).

How it works: Even though it has "regression" in the name, logistic regression is used for **classification**. It still computes a linear combination of inputs, but instead of outputting a direct value, it passes the result through a **sigmoid function** to map the output to a **probability between 0 and 1**.

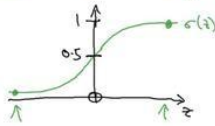
Equation:

Logistic Regression

Given x , want $\hat{y} = P(y=1|x)$
 $x \in \mathbb{R}^{d_x}$ $0 \leq \hat{y} \leq 1$

Parameters: $w \in \mathbb{R}^{d_x}$, $b \in \mathbb{R}$.

Output $\hat{y} = \sigma(\underbrace{w^T x + b}_z)$



$$x_0 = 1, \quad x \in \mathbb{R}^{d_x+1}$$

$$\hat{y} = \sigma(\theta^T x)$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{d_x+1} \end{bmatrix} \quad \left\{ \begin{array}{l} b \leftarrow \theta_0 \\ w \leftarrow \begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_{d_x+1} \end{bmatrix} \end{array} \right.$$

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

If z large $\sigma(z) \approx \frac{1}{1+0} = 1$
 If z large negative number $\sigma(z) = \frac{1}{1+e^{-z}} \approx \frac{1}{1+\text{Big num}} \approx 0$

Andrew Ng

Neural networks and basics of deep learning-

Neural Networks (NNs): Detailed Overview Definition:

Neural Networks are a class of **supervised learning algorithms** inspired by the structure of the human brain. They're composed of layers of simple computing units called **neurons** that learn to recognize patterns in data.

Core Structure:

1. **Input Layer:** Takes raw input data (e.g., pixels of an image, numeric features).
2. **Hidden Layers:** Perform feature extraction and transformation.
 - o Each neuron calculates a **weighted sum** of inputs, adds a **bias**, and applies an **activation function**.
3. **Output Layer:** Gives final prediction (e.g., class label, probability, or number).

Mathematical Operation (per neuron):

$$z = \sum_{i=1}^n w_i x_i + b; a = \text{activation}(z)$$

$$z = \sum_{i=1}^n w_i x_i + b; a = \text{activation}(z) \quad \text{Where:}$$

- x_i = input features
- w_i = weights
- b = bias
- a = output of the neuron after applying activation function **Activation**

Functions:

They introduce **non-linearity**, allowing the network to learn complex patterns.

- **Sigmoid:** $\frac{1}{1+e^{-x}}$ – used in binary classification
- **Tanh:** outputs in range $(-1, 1)$
- **ReLU:** $\max(0, x)$ – most popular in hidden layers due to simplicity and speed

What is Deep Learning?

Deep Learning refers to **neural networks with many layers**, capable of automatically extracting high-level features from raw data. These models **scale well with large data and computation power**.

A “shallow” neural network might have 1 hidden layer. A deep network could have dozens or hundreds of layers.

How Deep Learning Works (Training Process):

1. **Forward Pass:** Data moves layer by layer from input to output.
2. **Loss Calculation:** A loss function measures how wrong the prediction is (e.g., MSE for regression, Cross-Entropy for classification).
3. **Backpropagation:** Calculates gradients of the loss with respect to weights using the **chain rule**.
4. **Weight Update:** Weights are updated using **Gradient Descent** or optimizers like Adam.

Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a specialized type of deep neural network primarily used for image recognition and computer vision tasks. They are designed to automatically and adaptively learn spatial hierarchies of features from input images.

1. Core Components of a CNN

- **Convolutional Layer:**
The fundamental layer of a CNN performs the convolution operation. It

uses small filters (also known as kernels) that slide over the input image to extract local features such as edges, corners, and textures.

- **Stride:**
Stride refers to the number of pixels by which the filter moves across the input image. A larger stride reduces the output size and computation, while a stride of 1 ensures detailed feature capture.
- **Padding:**
Padding involves adding extra pixels (usually zeros) around the border of the input image. This helps preserve the spatial dimensions after convolution, especially when the filter size is greater than 1. Common types are '**valid**' (no padding) and '**same**' (output has same dimensions as input).
- **Activation Function (ReLU):**
After convolution, a non-linear activation function like ReLU (Rectified Linear Unit) is applied. It replaces negative values with zero, allowing the model to learn complex patterns and interactions.
- **Pooling Layer (e.g., Max Pooling):**
Pooling is used to reduce the spatial dimensions of feature maps, improving computational efficiency and controlling overfitting. **Max pooling** selects the maximum value in a region, helping retain the most prominent features.

2. Fully Connected Layer

After several convolution and pooling layers, the resulting feature maps are **flattened** into a 1D vector and passed through one or more **fully connected layers** (dense layers). These layers perform high-level reasoning and are typically responsible for the final classification or regression tasks.

3. Backpropagation in CNNs

Like other neural networks, CNNs are trained using **backpropagation**. During training:

- The model makes a prediction,
- The **loss** is calculated (using functions like cross-entropy),

- The error is propagated backward through the network using the **chain rule**,
- Weights (including filter values) are updated using **gradient descent** or its variants to minimize the loss.

This iterative process helps the CNN learn optimal filters and weights to improve accuracy.

4. CNN Implementation with Keras and TensorFlow

TensorFlow is an open-source machine learning library developed by Google. It provides low-level APIs for building and training deep learning models. On top of TensorFlow, **Keras** offers a high-level, user-friendly API for defining CNN architectures quickly and efficiently.