**Maximum Likelihood Estimation (MLE) Task**

NAME : TEJASRI .V

Maximum Likelihood Estimation (MLE) is a statistical method commonly used in generative AI models to estimate the parameters of a probabilistic model that best explains a given dataset. It's particularly important in generative models because these models aim to learn the underlying data distribution and generate new samples that are similar to the training data.

Here's how MLE works in the context of generative AI:

**Choose a Probability Distribution:** In generative AI, you start by assuming a specific probabilistic model that you believe generates the data. This could be a Gaussian distribution, a Bernoulli distribution, a mixture of Gaussians, or even a more complex model like a neural network-based generative model (e.g., GANs or VAEs).

**Define Likelihood Function:** The likelihood function describes how likely your chosen model is to generate the observed data. It's a function of the model parameters. For example, if you assume your data is Gaussian-distributed, the likelihood function would involve the mean and standard deviation of the Gaussian.

**Calculate Likelihood:** Calculate the likelihood of the observed data under the assumed probabilistic model. This involves plugging in the observed data into the likelihood function and evaluating how well the model explains the data.

**Maximize Likelihood:** The goal of MLE is to find the parameters that maximize the likelihood of the observed data. In mathematical terms, this often translates to finding the parameter values that make the observed data most probable under the model.

**Optimization:** Finding the parameters that maximize the likelihood might involve solving an optimization problem. Depending on the complexity of the model and the likelihood function, this optimization could be straightforward or quite challenging. Gradient-based optimization techniques are often used, and in the case of neural networks, backpropagation can be employed to compute gradients efficiently.

**Model Evaluation and Generation:** Once you've estimated the parameters using MLE, you have a trained generative model. You can use this model to generate new data samples that should resemble the original training data.

It's important to note that while MLE is a powerful method, it can sometimes lead to overfitting, especially in complex generative models. In such cases, more advanced techniques like regularization, Bayesian methods, or using alternative optimization objectives (as in Variational Autoencoders) might be employed.

Here's how MLE is applied in various AI models:

**Linear Regression:** In linear regression, MLE is used to estimate the coefficients (parameters) of the linear equation that best fits the observed data points. The goal is to find the parameters that maximize the likelihood of the observed target variable given the input features.

**Logistic Regression**: MLE is also used in logistic regression to estimate the parameters that best fit the sigmoid curve to the observed data, especially for binary classification problems. The likelihood is maximized to find the best-fitting curve.

**Naive Bayes Classifier:** In Naive Bayes classification, MLE is used to estimate the probabilities of different features given a class label. This estimation helps classify new data points into the most likely class based on observed feature probabilities.

**Hidden Markov Models (HMMs):** HMMs are used in various AI tasks, such as speech recognition and natural language processing. MLE is used to estimate the model parameters, such as transition probabilities and emission probabilities, that best explain the observed sequence of data.

**Gaussian Mixture Models (GMMs):** GMMs are often used for clustering and density estimation tasks. MLE is applied to estimate the parameters of Gaussian distributions that best explain the observed data points.

**Neural Networks:** In deep learning, MLE is used to train neural network models. When training a neural network, the goal is to find the model parameters that maximize the likelihood of the observed targets given the input data. This is often done using optimization algorithms like stochastic gradient descent (SGD) or its variants.

**Generative Adversarial Networks (GANs):** In GANs, MLE is used to train the generator network to produce samples that resemble the real data distribution. The generator aims to maximize the likelihood that the discriminator misclassifies its generated samples as real.

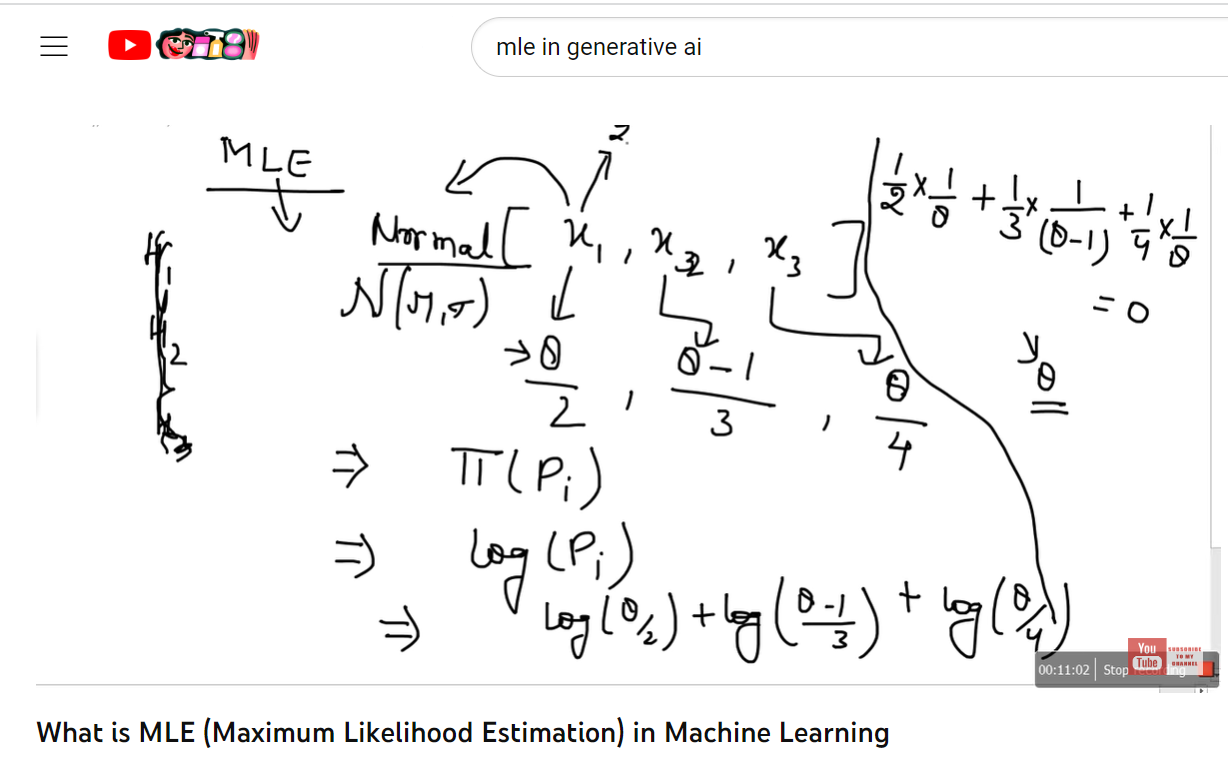
**Variational Autoencoders (VAEs):** VAEs use MLE to train the encoder and decoder networks. The goal is to maximize the evidence lower bound (ELBO), which involves maximizing the likelihood of generating observed data and minimizing a regularization term.

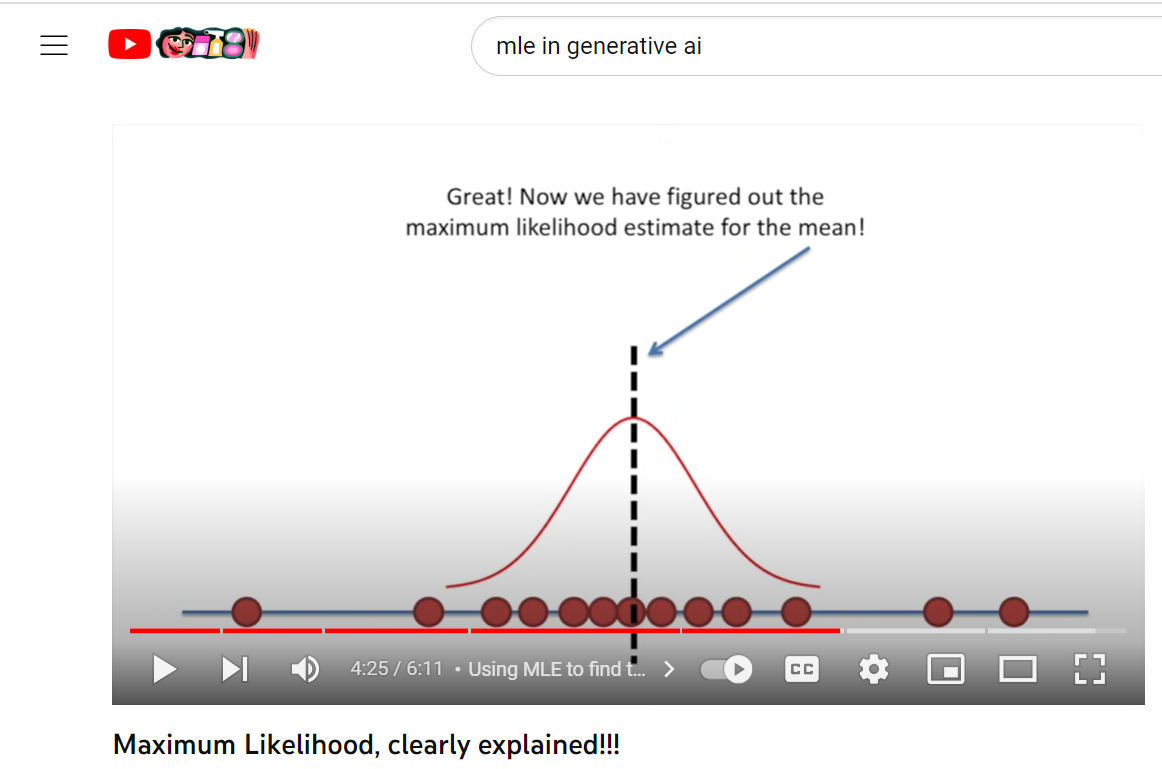
**Recurrent Neural Networks (RNNs):** In tasks like sequence generation and language modeling, RNNs use MLE to estimate the parameters that make the model generate sequences that match the observed sequences.

**Natural Language Processing:** MLE is used in various language modeling tasks, such as text generation, machine translation, and speech recognition, to estimate the parameters that capture the underlying patterns in the data.

In all these AI models, the central idea is to adjust the model parameters to make the observed data more probable according to the model's assumptions. MLE provides a principled way to achieve this by finding parameter values that maximize the likelihood of the observed data under the model.

LEARNT SOME CONCEPTS FROM YOUTUBE VIDEOS :





Here's a summary of the key points of the videos :

Likelihood and Maximum Likelihood Estimation:

* + The likelihood of observing a set of data given a certain parameterized probability distribution is the probability of observing that data under the assumed distribution.
  + Maximum Likelihood Estimation (MLE) is a method used to find the parameter values that maximize the likelihood of observing the given data.
  + In practice, it's common to work with the log-likelihood to simplify calculations.
  + The MLE of parameters is often obtained by solving the system of equations formed by setting the partial derivatives of the log-likelihood with respect to the parameters to zero.

Normal Distribution:

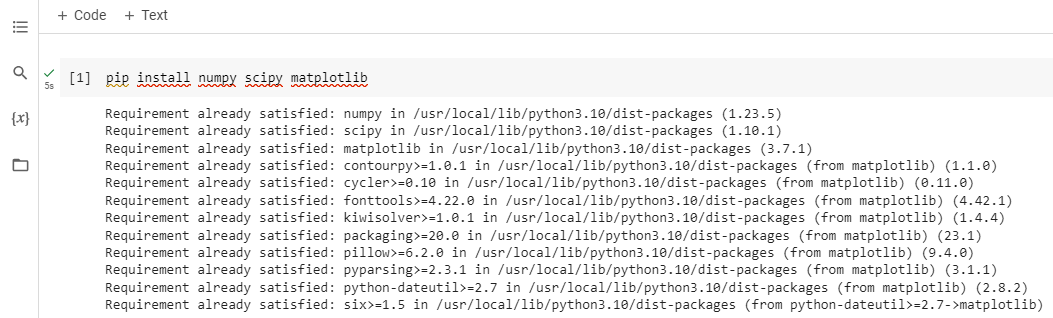
* + The normal (Gaussian) density function is defined by a mean (μ) and a standard deviation (σ).
  + The likelihood of observing a set of data points is the product of the individual likelihoods of each data point.

Optimization and Scipy:

* + You use Scipy's optimization functions (e.g., minimize ) to find the parameter values that maximize the log-likelihood.
  + Constraints can be added to optimization problems to ensure that the parameter values remain within certain bounds.

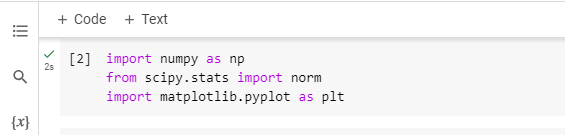
**Task:** Likelihood Maximum Estimation (MLE) of Gaussian Parameters with Python

* Install required libraries .



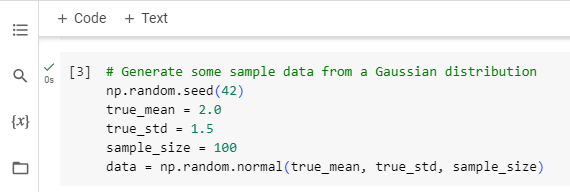
1. We import the necessary libraries:

'numpy' for numerical computations, 'scipy.stats.norm' for the Gaussian distribution functions, and matplotlib.pyplot for plotting.



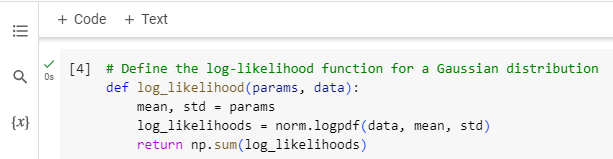
2. We set a random seed to ensure reproducibility of random numbers.

* 'true\_mean' and 'true\_std' are the true mean and standard deviation of the Gaussian distribution that we'll use to generate our sample data.
* 'sample\_size' determines the number of data points we generate.
* 'data' is an array of randomly generated data points sampled from the Gaussian distribution.

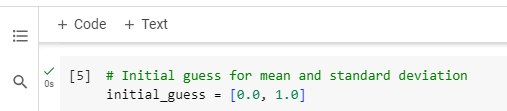


3. We define a function 'log\_likelihood' that calculates the log-likelihood of the given parameters 'params' (mean and standard deviation) for the observed 'data'.

* Inside the function, we calculate the log PDF (probability density function) of the data points using 'norm.logpdf' from the 'scipy.stats.norm' module.
* We sum up the log-likelihoods for all data points using 'np.sum'.

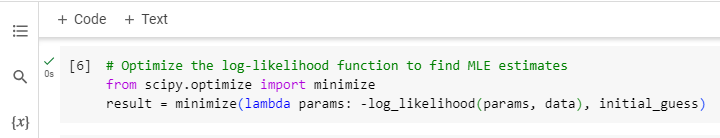


4. We provide an initial guess for the mean and standard deviation parameters.

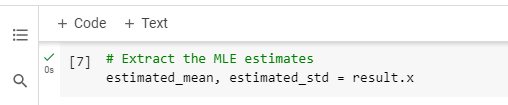


5. We use the 'minimize' function from 'scipy.optimize' to find the parameters that maximize the negative log-likelihood.

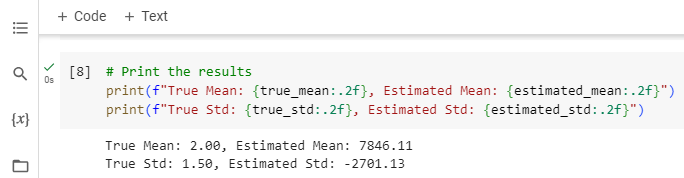
* The function to be minimized is defined as a lambda function. We use the negative log-likelihood '(-log\_likelihood(params, data)') because the 'minimize' function minimizes by default, but we want to maximize the likelihood.
* 'initial\_guess' is the starting point for the optimization.



6. After optimization, the 'result.x' attribute contains the estimated parameters that maximize the likelihood.



7. We print the true and estimated mean and standard deviation.



8. We create a histogram of the generated data using 'plt.hist' and overlay it with the estimated Gaussian distribution using 'norm.pdf'.

* We use 'plt.plot' to visualize the estimated Gaussian distribution.
* 'plt.legend()', 'plt.xlabel()', 'plt.ylabel()', 'plt.title()' set labels and titles for the plot and 'plt.show()' displays the plot.

