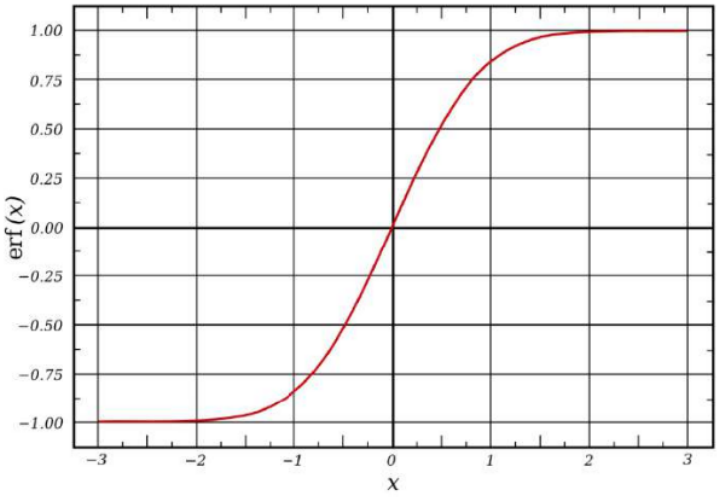
Module- 4

ERROR FUNCTIONS

Error function

In mathematics, the error function, commonly known as the Gauss error function and denoted by erf, is defined as erf=c->c.

$$\operatorname{erf} z = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}\, \mathrm{d}t.$$



The integral presented is a complex contour integral that is path-independent due to its holomorphic nature throughout the whole complex plane. In numerous applications, the function parameter is a real number, resulting in a real function value.

In several ancient manuscripts, the error function is defined absent the factor of . This non-elementary integral is a sigmoid function frequently encountered in probability, statistics, and partial differential equations.

Two closely associated functions are the

complementary error function
$\operatorname{erfc} : \mathbb{C} \to \mathbb{C}$ is defined as

$$\operatorname{erfc} z = 1 - \operatorname{erf} z,$$

and the **imaginary error function**
$\operatorname{erfi} : \mathbb{C} \to \mathbb{C}$ is defined as

$$\operatorname{erfi} z = -i \operatorname{erf} iz,$$

where i is the imaginary unit.

The term "error function" and its acronym erf were introduced by J. W. L. Glaisher in 1871 due to its association with "the theory of Probability, particularly the theory of Errors." The complementary error function was also addressed by Glaisher in a distinct book in the same year.

Applications

If a series of measurements follows a normal distribution with a standard deviation σ and an expected value of 0, then erf(a/σ √2) is the chance that the error of a single measurement falls between −a and +a, for positive a. This is beneficial, for instance, in ascertaining the bit error rate of a digital communication system.

The error and complementary error functions arise in solutions of the heat equation when boundary conditions are defined by the Heaviside step function.

Sum of Squares

The sum of squares refers to the aggregate of the squares of the specified numbers. In statistics, it refers to the summation of the squared deviations of a dataset. To accomplish this, we must calculate the mean of the data, determine the deviation of each data point from the mean, square these deviations, and sum them. In algebra, the sum of the squares of two numbers is calculated using the $(a + b)^2$ identity. The sum of squares of the first n natural numbers can alternatively be calculated using a formula. The formula can be established by the principle of mathematical induction. We perform fundamental arithmetic operations essential for statistics and algebra. Various approaches exist to calculate the sum of squares of specified numbers.

Sum of Squares

The sum of squares in statistics is a method employed to assess the dispersion of a dataset. To assess this, we calculate the sum of the squares of the deviations of each data point. In algebra, the sum of the squares of two numbers is determined using the algebraic identity $(a + b)^2$. In mathematics, the sum of the squares of n natural numbers is calculated using a specific formula derived from the principle of mathematical induction. We will now examine the formulas for calculating the sum of squares across many branches of mathematics.

Formula for the Sum of Squares

The sum of squares formula in statistics quantifies the degree to which the modeled data is accurately represented by a model. It illustrates the distribution of the dataset. The sum of squares formula is employed to compute the total of two or more squares in an equation. Consequently, several formulas for sums of squares are as follows:

•**In statistics:** The sum of squares of n data points is expressed as $\sum ni=0 (xi - \bar{x})^2$.

In algebra: The sum of squares is expressed as $a^2 + b^2 = (a + b)^2 - 2ab$.

Formula for the sum of squares of n natural numbers: $1^2 + 2^2 + 3^2 + ... n^2 = [n(n + 1)(2n + 1)] / 6$

In which location,

- $\sum$ denotes summation
- xi = individual value inside the set
- $\bar{x}$ = average of the values
- $xi - \bar{x}$ = departure from the mean
- $(xi - \bar{x})^2$ = square of the deviation
- a, b = arbitrary variables
- n = quantity of terms in the series

Let a and b represent the two numerical values. Let the squares of a and b be represented as a^2 and b^2. The sum of the squares of a and b is $a^2 + b^2$. We can derive a formula utilizing the established algebraic identity $(a+b)^2 = a^2 + b^2 + 2ab$. By subtracting 2ab from both sides, we may deduce that $a^2 + b^2 = (a + b)^2 - 2ab$. Let a, b, and c represent the three numbers for which we are to calculate the sum of their squares. The total of their squares is $a^2 + b^2 + c^2$. Utilizing the established algebraic

identity $(a+b+c)^2 = a^2 + b^2 + c^2 + 2ab + 2bc + 2ca$, we can ascertain that $a^2 + b^2 + c^2 = (a+b+c)^2 - 2ab - 2bc - 2ca$.

In statistics, the sum of squares error (SSE) represents the discrepancy between the observed value and the projected value. It is referred to as the sum of squares residual (SSR) because it represents the sum of the squares of the residuals, which are the deviations of predicted values from actual values. The equation for the sum of squares error is expressed as follows:

$SSE = \sum_{i=0}^{n} (y_i - f(x_i))^2$, where y_i represents the ith observation of the dependent variable, $f(x_i)$ denotes the predicted value, and x_i signifies the ith observation of the independent variable.

The sum of squares error (SSE) can be calculated by deducting the sum of squares regression (SSR) from the sum of squares total (SST), expressed as SSE = SST - SSR.

Important Notes on Sum of Squares

The sum of squares in statistics is a method employed to assess the variability of a dataset.

- SSE = SST - SSR
- The sum of squares of n data points is expressed as $\sum_{i=0}^{n} (x_i - \bar{x})^2$

Steps to Find Sum of Squares

The total sum of squares in statistics can be computed by the following steps:

- Step 1: Enumerate the data points in the dataset.
- Step 2: Compute the mean of the dataset.
- Step 3: Deduct each data point from the mean.
- Step 4: Calculate the square of the difference identified in step 3.
- Step 5: Incorporate the squares identified in step 4.

Examples of Sum of Squares

Example 1: Employing the sum of squares formula, determine the value of 42 + 62.

Objective: Determine the value of 42 + 62

Provided: a = 4, b = 6

Utilizing the sum of squares formula $a^2 + b^2 = (a + b)^2 - 2ab$, we obtain

$42 + 62 = (4 + 6)^2 - 2(4)(6)$

$100 - 2(24)$

100 minus 48

Equals 52

The sum of 42 and 62 is 52.

Example 2: Compute the total of the series 12 + 22 + 32 … 1002

Resolution:

Determine: The summation of the series

Applying the sum of squares formula for n terms, $12 + 22 + 32 + ... n^2 = [n(n+1)(2n+1)] / 6$

Provided: n = 100

$= [100(101)(201)] / 6$

$(100 \times 101 \times 201) / 6$

Equals 338350

The total of the specified series is 338350.

Minkowski Distance

The Minkowski distance, or Minkowski metric, is a metric in a normed vector space that generalizes both the Euclidean distance and the Manhattan distance. It is named in honor of the Polish mathematician Hermann Minkowski.

Definition

The Minkowski distance of order p (where p is an integer) between two points

$$X = (x_1, x_2, \ldots, x_n) \text{ and } Y = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$$

is defined as:

$$D(X, Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}.$$

For $p \geq 1$, the Minkowski distance constitutes a metric according to the Minkowski inequality. When $p < 1$, the distance between (0,0) and (1,1) is undefined, although the point (0,1) is equidistant from both of these positions. This does not satisfy the triangle inequality; hence, it is not a metric. Nevertheless, a metric can be derived for

these values by only eliminating the exponent. The resultant metric is similarly classified as an F-norm.

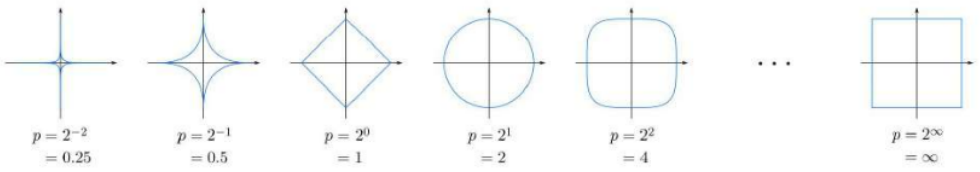
The Minkowski distance is generally utilized with values of 1 or 2, representing the Manhattan distance and the Euclidean distance, respectively. In the extreme scenario where the value approaches infinity, the Chebyshev distance is derived.

$$\lim_{p\to\infty}\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{\frac{1}{p}}=\max_{i=1}^{n}|x_i-y_i|.$$

Similarly, for p reaching negative infinity, we have:

$$\lim_{p\to-\infty}\left(\sum_{i=1}^{n}|x_i-y_i|^p\right)^{\frac{1}{p}}=\min_{i=1}^{n}|x_i-y_i|.$$

The Minkowski distance can also be interpreted as a multiple of the power mean of the component-wise discrepancies between P and Q.



Applications

The Minkowski metric is highly beneficial in the domains of machine learning and artificial intelligence. Numerous prevalent machine learning algorithms employ particular distance metrics, as previously indicated, to assess the similarity between two data points. Various metrics can be employed based on the characteristics of the data under analysis. The Minkowski metric is particularly advantageous for numerical datasets when assessing the similarity in magnitude among various data point vectors.

Input dependent variance

Input-dependent variance is a notion applicable in various situations, including signal processing, wireless signal strength, and regression analysis.

Conditional probability distribution

In probability theory and statistics, the conditional probability distribution delineates the likelihood of a result contingent upon the occurrence of a specific event. For two jointly distributed random variables Y and X, the conditional probability distribution of X given Y represents the probability distribution of X when Y is fixed at a specific value; in certain instances, the conditional probabilities may be articulated as functions with the unspecified value x of X as a parameter. A conditional probability table is generally employed to illustrate the conditional probability when both X and Y are categorical variables. The conditional distribution differs from the marginal distribution of a random variable, which represents its distribution independent of the value of the other variable.

If the conditional distribution of Y given X is continuous, its probability density function is termed the conditional density function. The characteristics of a conditional distribution, including moments, are commonly designated by corresponding terms such as conditional mean and conditional variance.

More broadly, one can denote the conditional distribution of a subset from a set comprising multiple variables; this conditional distribution depends on the values of all other variables, and if the subset contains multiple variables, it represents the conditional joint distribution of those variables.

Conditional discrete distributions

$$p_{Y|X}(y \mid x) \triangleq P(Y = y \mid X = x) = \frac{P(\{X = x\} \cap \{Y = y\})}{P(X = x)}$$

Due to the occurrence of $P(X = x)$ in the denominator, this is defined only for non-zero (hence strictly positive) $P(X = x)$.

The relation with the probability distribution of X given Y is:

$$P(Y = y \mid X = x)P(X = x) = P(\{X = x\} \cap \{Y = y\}) = P(X = x \mid Y = y)P(Y = y).$$

The conditional probability mass function of Y given X=x for discrete random variables can be expressed as per its definition:

Example

Consider the roll of a fair die and let $X = 1$ if the number is even (i.e., 2, 4, or 6) and $X = 0$ otherwise. Furthermore, let $Y = 1$ if the number is prime (i.e., 2, 3, or 5) and $Y = 0$ otherwise.

D	1	2	3	4	5	6
X	0	1	0	1	0	1
Y	0	1	1	0	1	0

Then the unconditional probability that $X = 1$ is 3/6 = 1/2 (since there are six possible rolls of the dice, of which three are even), whereas the probability that $X = 1$ conditional on $Y = 1$ is 1/3 (since there are three possible prime number rolls—2, 3, and 5—of which one is even).

Relation to independence

Random variables X, Y are independent if and only if the conditional distribution of Y given X is, for all possible realizations of X, equal to the unconditional distribution of Y. For discrete random variables this means $P(Y = y|X = x) = P(Y = y)$ for all possible y and x with $P(X = x) > 0$. For continuous random variables X and Y, having a joint density function, it means $f_Y(y|X = x) = f_Y(y)$ for all possible y and x with $f_X(x) > 0$.

Posterior probability

The posterior probability is a conditional probability derived from revising the prior probability using information encapsulated by the likelihood through the use of Bayes' theorem. From an epistemological standpoint, the posterior probability encompasses all knowledge regarding an uncertain proposition (such as a scientific hypothesis or parameter values), contingent upon past knowledge and a mathematical model that delineates the observations accessible at a specific time. Upon receiving new information, the existing posterior probability may function as the prior in a subsequent round of Bayesian updating.

Definition in the distributional case

In Bayesian statistics, the posterior probability represents the likelihood of the parameters contingent upon the data and is represented as such.

Given a prior belief that a probability distribution function is $p(\theta)$ and that the observations x have a likelihood $p(x|\theta)$, then the posterior probability is defined as

$$p(\theta|x) = \frac{p(x|\theta)}{p(x)}p(\theta),$$[6]

where $p(x)$ is the normalizing constant and is calculated as

$$p(x) = \int p(x|\theta)p(\theta)d\theta$$

The posterior probability is thus proportional to the product of likelihood and prior probability.

Example

The event G is that the student observed is a girl, and the event T is that the student observed is wearing trousers. To compute the posterior probability $P(G|T)$, we first need to know:

- $P(G)$, or the probability that the student is a girl regardless of any other information. Since the observer sees a random student, meaning that all students have the same probability of being observed, and the percentage of girls among the students is 40%, this probability equals 0.4.
- $P(B)$, or the probability that the student is not a girl (i.e. a boy) regardless of any other information (B is the complementary event to G). This is 60%, or 0.6.
- $P(T|G)$, or the probability of the student wearing trousers given that the student is a girl. As they are as likely to wear skirts as trousers, this is 0.5.
- $P(T|B)$, or the probability of the student wearing trousers given that the student is a boy. This is given as 1.
- $P(T)$, or the probability of a (randomly selected) student wearing trousers regardless of any other information. Since $P(T) = P(T|G)P(G) + P(T|B)P(B)$ (via the law of total probability), this is $P(T) = 0.5 \times 0.4 + 1 \times 0.6 = 0.8$.

Given all this information, the **posterior probability** of the observer having spotted a girl given that the observed student is wearing trousers can be computed by substituting these values in the formula:

$$P(G|T) = \frac{P(T|G)P(G)}{P(T)} = \frac{0.5 \times 0.4}{0.8} = 0.25.$$

Consider a school where the student population comprises 60% boys and 40% girls. The girls don trousers or skirts in equal proportions; all guys don pants. An observer perceives a pupil from afar, noting solely that the student is attired in trousers. What is the likelihood that this student is female? The accurate solution can be determined by Bayes' theorem.

Calculation

The posterior probability distribution of one random variable, conditioned on the value of another, can be computed using Bayes' theorem by multiplying the prior probability distribution by the likelihood function and subsequently dividing by the normalizing constant, as demonstrated:

$$f_{X|Y=y}(x) = \frac{f_X(x)\mathcal{L}_{X|Y=y}(x)}{\int_{-\infty}^{\infty} f_X(u)\mathcal{L}_{X|Y=y}(u)\,du}$$

gives the posterior probability density function for a random variable X given the data $Y = y$, where

- $f_X(x)$ is the prior density of X,
- $\mathcal{L}_{X|Y=y}(x) = f_{Y|X=x}(y)$ is the likelihood function as a function of x,
- $\int_{-\infty}^{\infty} f_X(u)\mathcal{L}_{X|Y=y}(u)\,du$ is the normalizing constant, and

Cross-entropy for classification

Literature on computer vision and deep learning typically recommends employing binary_cross entropy for binary (two-class) problems and categorical_cross entropy for scenarios involving several classes.

categorical cross-entropy:

It accepts solely one accurate classification per sample. Will exclusively utilize the real neuron to perform the cross-entropy computation with that neuron.

Binary cross-entropy:

• permits several accurate classifications per sample.

• Will perform the cross-entropy calculation for "all neurons," acknowledging that each neuron can belong to one of two classes: 0 or 1.

A binary classification problem can be represented as:

• Two-neuron output with a single accurate class: softmax combined with categorical cross entropy

• Single-neuron output, one class represented as 0 and the other as 1: sigmoid activation with binary cross-entropy loss.

Explanation

$$CE = -\sum_{neuron=1}^{classes} y_{true_{neuron}} * \ln\left(y_{pred_{neuron}}\right)$$

$$BCE_{1_{neuron}} = -\left[y_{true} * \ln(y_{pred}) + (1 - y_{true}) * \ln\left(1 - y_{pred}\right)\right]$$

Observe that binary cross entropy (the second equation in the image) comprises two components: one for treating 1 as the correct class and another for treating 0 as the correct class.

Categorical Cross-Entropy in Multi-Class Classification

Categorical Cross-Entropy (CCE), referred to as softmax loss or log loss, is a prevalent loss function in machine learning, especially for classification tasks. It quantifies the disparity between the anticipated probability distribution and the actual distribution of classes. The function assists a machine learning model in assessing the deviation of its predictions from the actual labels and directs it in enhancing prediction accuracy.

Understanding Categorical Cross-Entropy

Categorical cross-entropy is employed in classification problems involving many classes (multi-class classification). It quantifies the disparity between two probability distributions: the expected distribution and the actual distribution, represented by a one-hot encoded vector.

In a one-hot encoded vector, the accurate class is denoted by "1," while all other classes are indicated by "0." Categorical cross-entropy penalizes predictions according to the model's confidence in the correct class.

Mathematical Formulation of Categorical Cross-Entropy

$L(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$

Where:

- $L(y, \hat{y})$ is the categorical cross-entropy loss.
- y_i is the true label (0 or 1 for each class) from the one-hot encoded target vector.
- $\hat{y}_i$ is the predicted probability for class i.
- C is the number of classes.

Computing Categorical Cross-Entropy

We will analyze the calculation of categorical cross-entropy through a mathematical example utilizing the subsequent true labels and anticipated probability.

We possess three samples, each corresponding to one of three categories (Class 1, Class 2, or Class 3). The actual labels are encoded in a one-hot format.

One.Actual Labels (y_true):

Example 1: Class 2 → [0, 1, 0]

True Labels (y_true):

Example 1: Class 2 → [0, 1, 0]

Example 2: Class 1 → [1, 0, 0]

Example 3: Class 3 → [0, 0, 1]

Predicted Probabilities (y_pred):

Example 1: [0.1, 0.8, 0.1]

Example 2: [0.7, 0.2, 0.1]

Example 3: [0.2, 0.3, 0.5

Example 1: True Label `[0, 1, 0]`, Predicted `[0.1, 0.8, 0.1]`

The true class is Class 2, so $y_2 = 1$, and we focus on the predicted probability for Class 2, which is $\hat{y}_2 = 0.8$.

$$L_1 = -\left(0 \cdot \log(0.1) + 1 \cdot \log(0.8) + 0 \cdot \log(0.1)\right)$$

Simplifying:

$$L_1 = -\log(0.8) = -(-0.22314355) = 0.22314355$$

Example2: Category 1 → [1, 0, 0]

Example 2: True Label `[1, 0, 0]`, Predicted `[0.7, 0.2, 0.1]`

The true class is Class 1, so $y_1 = 1$, and we focus on the predicted probability for Class 1, which is $\hat{y}_1 = 0.7$.

$$L_2 = -\left(1 \cdot \log(0.7) + 0 \cdot \log(0.2) + 0 \cdot \log(0.1)\right)$$

Simplifying:

$$L_2 = -\log(0.7) = -(-0.35667494) = 0.35667494$$

Example 3: True Label `[0, 0, 1]`, Predicted `[0.2, 0.3, 0.5]`

The true class is Class 3, so $y_3 = 1$, and we focus on the predicted probability for Class 3, which is $\hat{y}_3 = 0.5$.

$$L_3 = -\left(0 \cdot \log(0.2) + 0 \cdot \log(0.3) + 1 \cdot \log(0.5)\right)$$

Categorical Cross-Entropy Works

1. **Prediction of Probabilities**. These probabilities represent the possibility of a data point being classified into each category. This is generally accomplished by a softmax function, which transforms raw scores into probabilities.
2. **Comparison with True Class** entropy evaluates the predicted probability against the true class labels (one-hot encoded).
3. **Calculation of Loss** computed by taking the logarithm of the predicted probability for the correct class, with the loss function imposing a penalty on the model proportional to the deviation of the forecast from the actual class.

Application of Categorical Cross-Entropy in Multi-Class Classification

Categorical cross-entropy is crucial in multi-class classification, when a model must assign an instance to one of multiple classes. In an image classification challenge, the model must determine whether an image depicts a cat, dog, or bird. CCE assists the model in modifying its weights throughout training to enhance predictive accuracy.

The CCE loss function presupposes that each data point is assigned to a singular class. In scenarios where a data point may simultaneously belong to many classes, binary cross-entropy is the preferred option.

Differences Between Categorical and Binary Cross-Entropy

Binary and categorical cross-entropy are employed to compute loss in classification tasks, although they vary in their applications and methods for managing numerous classes:

Binary Cross-Entropy is employed for binary classification tasks that have two potential outcomes (e.g., "yes" or "no").

Categorical Cross-Entropy is employed for multi-class classification including three or more categories, wherein the model allocates probability to each category.

Hyper parameters Optimization methods

Hype parameters are the parameters established for the training process. Hyper parameters significantly influence both accuracy and efficiency during model training. Consequently, it required precise calibration to get optimal and efficient outcomes. Let us examine various methods for hyperparameter optimization.

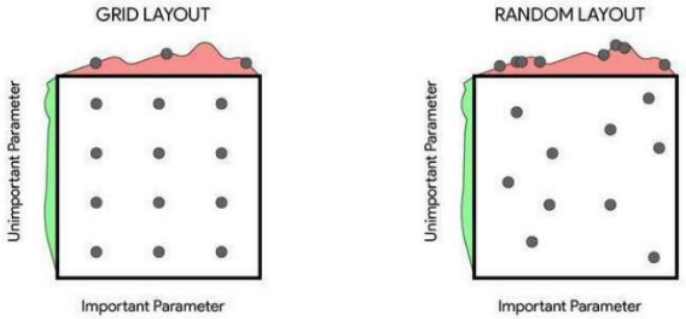
Hyperparameters Optimization Technique

Exhaustive Search Methods

Initially, let us examine several Exhaustive Search Methods to optimize hyper parameters.

• **Grid Search:** In Grid Search, the potential values of hyper parameters are specified within a specific set. The sets of potential hyper parameter values are joined using the Cartesian product to create a multidimensional grid. Subsequently, we evaluate all parameters within the grid and identify the hyper parameter configuration that yields the optimal outcome.

• **Random Search:** This method is a version of Grid Search that involves selecting random points rather than evaluating all points within the grid. This addresses several issues inherent in Grid Search, notably the necessity to exponentially increase the search space with the addition of each new hyper parameter.



Drawback:

Random Search and Grid Search are straightforward to build and can operate concurrently; nonetheless, they possess certain limitations.

• A substantial hyper parameter search space necessitates considerable time and computer resources for optimization.

• These methods do not ensure the identification of local maxima if the sampling is not conducted with precision.

Bayesian Optimization:

In Bayesian optimization, we utilize prior information to estimate the hyper parameters rather than relying on random guesses. They utilize these data to construct a probabilistic model that correlates hyper parameters with a probability function of a score on the objective function. The probability function is defined below.

$$P\left(\frac{score(y)}{hyperparameters(x)}\right)$$

This function is also referred to as the "surrogate" of the objective function. It is significantly simpler to optimize than the objective function. The following are the procedures for implementing Bayesian Optimization for hyper parameter optimization:

1. Construct a surrogate probability model for the objective function.
2. Identify the hyper parameters that yield optimal performance on the surrogate.
3. Implement these hyper parameters in the original objective function.
4. Revise the surrogate model with the recent findings.
5. Reiterate steps 2–4 until the specified number of iterations is achieved.

Alternative Hyper parameter Estimation Algorithms:

Hyperband:

The fundamental premise of this algorithm posits that if a hyper parameter configuration is likely to be optimal after numerous iterations, it is more probable to rank in the upper half of configurations after a limited number of iterations. The following outlines the step-by-step implementation of Hyperband.

1. Select a random sample of n hyper parameter configurations from the search space.
2. After k iterations, assess the validation loss of these hyper parameters.
3. Eliminate the lowest performing half of the hyper parameters.
4. Execute the successful candidates for an additional k iterations, assess their performance, and eliminate the lowest half.
5. Continue iterating until just a single hyperparameter model remains.

Drawback:

In the case of a high sample size, well-performing hyper parameter sets that require significant time to converge may be prematurely eliminated during the optimization process.

Population-Based Training (PBT):

Population-based Training (PBT) commences in a manner akin to random-based training, involving the simultaneous training of multiple models. Instead of allowing the networks to train independently, it utilizes data from the rest of the population to optimize hyper parameters and allocate computational resources to promising models. This draws inspiration from genetic algorithms, wherein each individual in the population, termed a worker, can utilize knowledge from the other members of the population. For example, an employee may replicate the model parameters from a significantly more proficient colleague. It can also investigate new hyper parameters by randomly altering the current values.

Bayesian Optimization and Hyper Band (BOHB):

BOHB (Bayesian Optimization and Hyper Band) integrates the Hyperband technique with Bayesian optimization. Initially, it employs Hyperband functionality to sample numerous configurations with a limited budget, facilitating a rapid and efficient exploration of the hyper-parameter search space to swiftly identify promising configurations. Subsequently, it utilizes the predictive capabilities of Bayesian

optimization to suggest a set of hyper parameters that are near optimal. This algorithm can also be executed in parallel, akin to Hyperband, thereby mitigating a significant limitation of Bayesian optimization.

Gradient Descent Method

A gradient is a derivative that delineates the impact on a function's outputs resulting from minor variations in its inputs.

Gradient Descent

Gradient Descent is fundamental to the complex process of model optimization. Fundamentally, it is a numerical optimization procedure designed to identify the ideal parameters—weights and biases—of a neural network by minimizing a specified cost function.

Gradient Descent (GD) is a prevalent optimization technique in machine learning and deep learning that minimizes the cost function of a neural network model during training. The process involves systematically modifying the model's weights or parameters in accordance with the negative gradient of the cost function until the cost function's minimum is attained.

Gradient Descent is a principal optimization approach in machine learning employed to minimize the cost or loss function during model training.

It incrementally modifies model parameters by progressing towards the greatest decline in the cost function.

The algorithm computes gradients, which denote the partial derivatives of the cost function with respect to each parameter.

Gradient Descent Implementation

Import the necessary libraries

```
import torch
import torch.nn as neural_network
import matplotlib.pyplot as plt
```

Configure the input and output data

Establish random seed for reproducibility.

```
torch.manual_seed(42)
```

Specify the quantity of samples.

```
number_of_samples = 1000
```

Create random features with two dimensions.

```
x = torch.randn(num_samples, 2)
```

Generate random weights and bias for the linear regression model.

true_weights = torch.tensor([1.3, -1.0])

true_bias = torch.tensor([-3.5])

Target Variable

y = x @ true_weights^T + true_bias

Visualize the dataset

fig, ax = plt.subplots(1, 2, sharey=True)

ax[0].scatter(x[:, 0], y)

ax[1].scatter(x[:, 1], y)

ax[0].set_x label('X1')

ax[0].set_y label('Y-axis Label')

ax[1].set_x label('X2')

ax[1].set_y label('Y-axis Label')

Display the plot.