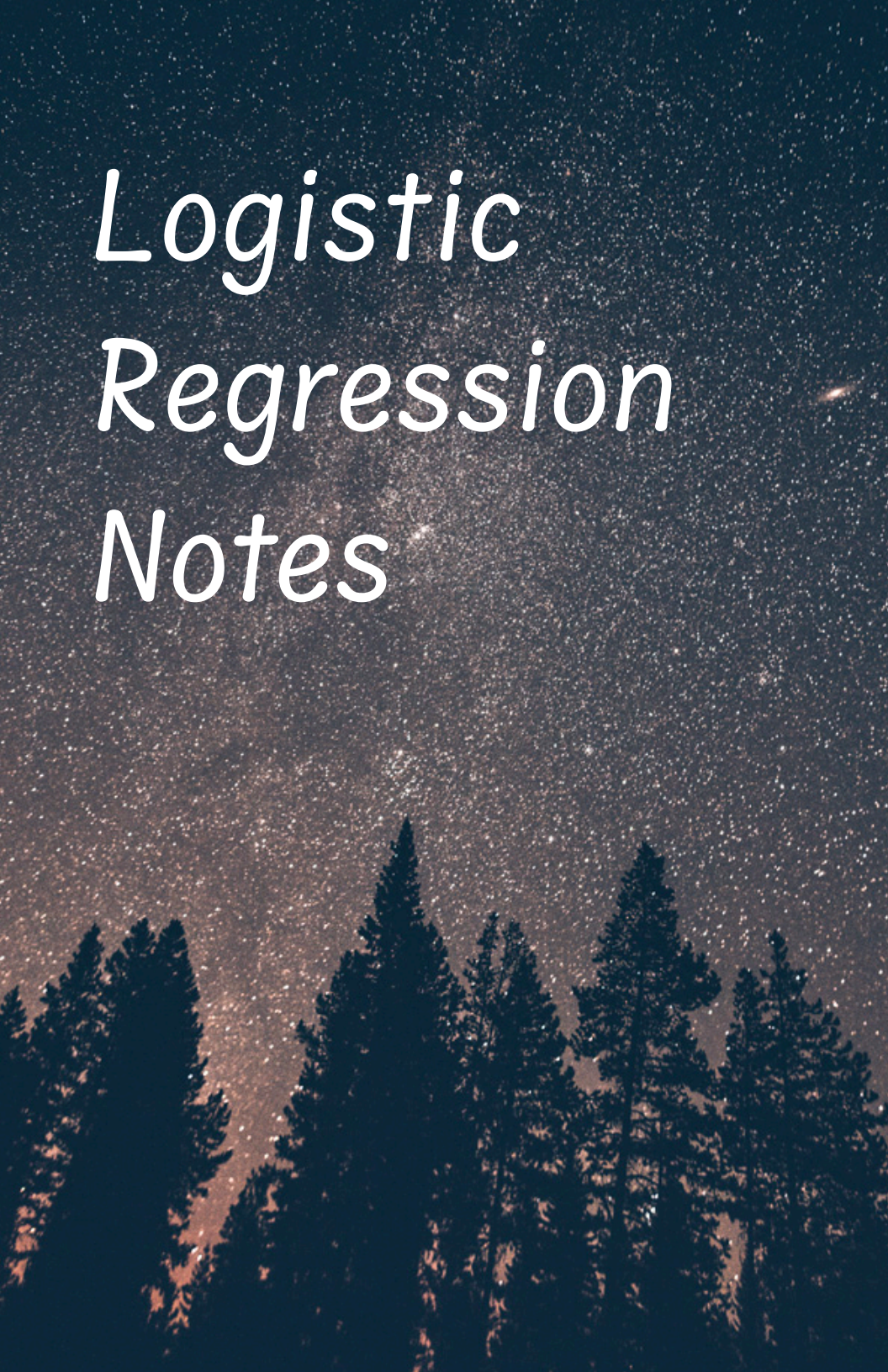
Logistic Regression Notes



Logistic Regression

Overview

Difference between Linear and Logistic Regression

1. Probabilistic

1.1 Perceptron

1.2 Sigmoid Function

1.3 Loss Function

1.4 Accuracy and Confusion Matrix

2 Softmax Regression

3 Polynomial Logistic Regression

4 Geometric

Overview

Logistic Regression is a classification algorithm which finds the probabilities of the data points to belong to a default class. We can understand Logistic Regression by Geometry, Probability, and loss function based interpretation. Logistic Regression is simply a classification technique whose task to find a hyperplane (n-Dimensional) or line (2-D) that best separates the classes.

Why does it has regression in its name given it is classifying ?

The probabilities are calculated by using a non linear logistic function also called it sigmoid function since these probability values are between 0 and 1 continues number so it has a name regression. We can finally covert these probability values into class labels by applying threshold.

Types

1. Binary Logistic Regression
2. Multinomial Logistic Regression
3. Ordinal Logistic Regression

Intuition

Our goal is to model the probability that a given input which belongs to a particular class.

1. *From a probabilistic standpoint. The sigmoid function is applied to the linear combination of input features. This transformation maps the output to probabilities between* 0 *and* 1*, representing the likelihood of a data point belonging to a particular class. By modelling these probabilities, logistic regression provides a probabilistic interpretation of the classification task, allowing for uncertainty estimation and confidence in predictions. Therefore, logistic regression is often referred to as a probabilistic method, as it directly models the probabilities of class membership.*

2. *From a geometric standpoint, logistic regression aims to find a decision boundary in the feature space that effectively separates different classes. This decision boundary is represented by a hyperplane, defined by the coefficients of the logistic regression model. By iteratively adjusting these coefficients, typically through techniques like gradient descent, logistic regression optimises the decision boundary to best fit the training data. The geometric intuition behind logistic regression lies in finding this optimal hyperplane that maximises the margin between classes while minimising classification errors, ultimately leading to effective class separation.*

Difference between Linear Regression and Logistic Regression

Linear regression is used for predicting continuous outcomes and Logistic regression is used for predicting the probability of a categorical outcome.

The dependent variable in linear regression is continuous, meaning it can take any real value within a given range. Where the dependent variable in logistic regression is categorical, typically representing binary outcomes or multinomial outcomes.

In linear regression, the model produces continuous predictions that can take any real value. In logistic regression, the model produces probabilities of belonging to a specific category, which are often converted into class predictions based on a chosen threshold.

1. Probabilistic

In logistic regression, the probabilistic approach forms the base of predictive modelling. Rather than providing definitive outcomes, logistic regression give probabilities, making it suitable for classification tasks. By using the sigmoid function, logistic regression transforms raw inputs into probability estimates, facilitating the prediction of categorical outcomes. This probabilistic, joined with techniques like maximum likelihood estimation and appropriate loss functions, enables logistic regression to effectively model the likelihood of events and make informed classifications.

1.1 Perceptron

The Perceptron method is not widely used in logistic regression. Instead, logistic regression commonly use the sigmoid function to model the probability of a binary outcome. The perceptron algorithm is an older method for binary classification that directly applies a linear combination of input features and step function to make predictions.

It is not accurate as sigmoid function but the reason of learning perceptron method is it share some similarities. It create a base foundation of sigmoid function. Even both methods involve learning weights for input features to make predictions.

One things that we should keep in mind perceptron algorithm can be effective for simple binary classification tasks.

Equation

$$\sum_{i=1}^{n} x_i \cdot w_i + b$$

Here, x_i represents the input features, w_i represents the corresponding weights b is the bias term. This equation helps to find that line, which separates classes in data.

Explanation

$$w_{\text{new}} = w_{\text{old}} + \eta \cdot (y - \hat{y}) \cdot X$$

In above formula w is a weight vector, η is learning rate, y is true label selected data point, $\hat{y}$ is predicted label, X is feature vector of the selected data point. We would find $\hat{y}$ using this formula $\hat{y} = \text{step}(w \cdot X)$ where w is weight vector and X is feature vector and we are passing product of w and X.

Step function

$$\text{step}(z) = \begin{cases} 1 & \text{if } z > 0 \\ 0 & \text{if } z \leq 0 \end{cases}$$

The step function is used to convert the continuous output of the dot product between the feature vector and the weights vector into a binary classification. If the output is positive or zero, the step function returns 1, indicating one class, while if the output is negative, it returns 0, indicating the other class. This binary output is then compared with the true label to compute the classification error and update the weights accordingly during the training process.

Training Process

The algorithm iterates for a fixed number of iterations, specifically n times. This fixed number of iterations is a hyper-parameter chosen by the user and can be adjusted based on the specific problem and dataset. Each iteration involves randomly selecting a sample from the dataset, computing the predicted output using the current weights, updating the weights based on the prediction error, and repeating the process. This

iterative process continues until the specified number of iterations is reached. Adjusting the number of iterations can affect the convergence and performance of the perceptron algorithm on the given task.

Return Values to Decision Boundary Line

Function will return w *is a weight vector. To calculate the slope* m *and the intercept* b *of the decision boundary from the* w *weight vector.* $b = -\frac{\mathrm{w0}}{\mathrm{w2}}$ $m = -\frac{\mathrm{w1}}{\mathrm{w2}}$ *These formulas represent the slope and intercept of the decision boundary line* $y = m\,x + b$ *in the context of a 2D plane where* x *and* y *are the features and* m *and* b *are the slope and intercept, respectively.*

1.2 Sigmoid Function

The sigmoid function is commonly used in logistic regression to model the probability of a binary outcome.

Formula

The sigmoid function is also known as the logistic function and is defined as $\sigma(x) = \frac{1}{1 + e^{-x}}$ where z is the input variable. This function maps any real-valued number to the range $(0, 1)$ which makes it suitable for modelling probabilities or binary classification tasks. When x is large and positive, e^{-x} approaches 0, causing to $\sigma(x)$ to approach 1 but when x is large and negative, e^{-x} become very large, causing to $\sigma(x)$ to approach 0. This sigmoid function has an S-shaped curve, with its midpoint at $x = 0$, where $\sigma(x) = 0.5$.

The output of the sigmoid function always lies between 0 and 1, and its S-shaped curve.

The sigmoid function approaches 0 as x approaches negative infinity, and approaches 1 as x approaches positive infinity.

In logistic regression, the output of the sigmoid function represents the probability that a given input belongs to a particular class.

Training Process and Return Values

Functions working and Return Value to Decision Boundary Line is same as defined in perceptron section but small change is we will replace step function to sigmoid function.

Magnitude

In contrast to the perceptron algorithm, which updates weights only when the predicted and true labels conflict, the sigmoid function employed in logistic regression facilitates continuous adjustments to weights during training. This ensures that every iteration contributes to refining the model, gradually optimising it towards better performance. The sigmoid's continuous nature allows for subtle adjustments in weight values, influencing the model's output probabilities without suddenly changes. This iterative process enables logistic regression to capture complex relationships in the data to an optimal solution.

1.3 Loss function

In logistic regression, the loss function, also known as the error function or cost function, quantifies the discrepancy between the predicted probabilities and the actual class labels. One common loss function used in logistic regression is the binary cross-entropy loss function. It measures the difference between the predicted probability of the positive class and the true label (which is either 0 or 1 for binary classification). The binary cross-entropy loss penalises misclassifications more severely when the predicted probability diverges significantly from the true label, encouraging the model to make accurate probability estimates. The goal during training is to minimise this loss function, typically achieved using optimisation techniques like gradient descent. Minimising the loss function results in a logistic regression model that effectively discriminates between different classes and accurately predicts probabilities.

Maximum Likelihood Estimation

Maximum Likelihood Estimation is a statistical method used to estimate the parameters (coefficients) of the logistic regression model. The key idea behind Maximum Likelihood Estimation is to find the parameter values that maximise the likelihood function, which represents the probability of observing the given data under the assumed statistical model.

Cross-entropy Loss Function

The cross-entropy loss function measures the dissimilarity between the true probability distribution of the data and the predicted probability distribution produced by the model. In the case of logistic regression, where the goal is to predict binary outcomes like 0 or 1, the cross-entropy loss function quantifies the difference between the true labels and the predicted probabilities assigned by the model.

Formula

$$L(y,\ \hat{y}) = \frac{-1}{n}\sum_{i=1}^{n}[yi\ \log(\hat{y}i)) + (1 - yi)\ \log(1 - \hat{y}i)]$$

y represents the true labels like 0 or 1, yi represents the predicted probabilities produced by the model. $\hat{y}i$ is the number of samples in the dataset and it will find with sigmoid function.

Gradient Discent

The role of gradient descent in logistic regression is essential for optimising the model parameters (coefficients) to minimise the loss function, such as the cross-entropy loss. Gradient descent is an iterative optimisation algorithm that updates the parameters in the direction of the steepest descent of the loss function.

Formula

$$\beta_{\text{new}} = \beta_{\text{old}} - \alpha \cdot \nabla J(\beta)$$

This formula is for update the coefficients in the opposite direction of the gradient to move towards the minimum of the loss function. α represents the learning rate, $\nabla J(\beta)$ is the gradient of the loss function.

Derivative of Sigmoid Function

$$\frac{d\sigma(x)}{dx} = \sigma(x) \cdot (1 - \sigma(x))$$

$\sigma(x)$ *is defined as* $\frac{1}{1 + e^{-x}}$

Derivative of the Cross-Entropy Loss Function

$$\frac{\partial J(\beta_j)}{\partial \beta_j} = -\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i) \cdot x_{ij}$$

$J(\beta)$ *represents the cross-entropy loss function.* β *is the vector of coefficients.* y *represents the actual label (*0 *or* 1*) of the* $i-th$ *observation.* x_{ij} *represents the* $j-th$ *feature of the* $i-th$ *observation.*

1.4 Accuracy and Confusion Matrix

Accuracy

Accuracy is a common metric used to measure the overall performance of a classification model. It represents the ratio of correctly predicted observations to the total number of observations in the dataset.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

In scikit-learn accuracy_score() function is for calculating accuracy.

Confusion Matrix

	Predicted Class	
Actual Class	**Positive**	**Negative**
Positive	**True Positive**	**False Negative**
Negative	**False Positive**	**True Negative**

True Positive (TP): The model correctly predicts the positive class.

True Negative (TN): The model correctly predicts the negative class.

False Positive (FP): The model incorrectly predicts the positive class (Type I error).

False Negative (FN): The model incorrectly predicts the negative class (Type II error).

In scikit-learn confusion_matrix() function.

In multi-classification, below one example explained.

	Predicted Class		
Actual Class	Class A	Class B	Class C
Class A	TP(A)	FP(A, B)	FP(A, C)
Class B	FP(B, A)	TP(B)	FP(B, C)
Class C	FP(C, A)	FP(C, B)	TP(C)

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

Precision

It focuses on the accuracy of positive predictions, indicating how many of the predicted positive instances are actually positive.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall

It focuses on capturing all positive instances, indicating how many of the actual positive instances were correctly identified by the model.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1 Score

F1 score is particularly useful when you want to consider both false positives and false negatives, and you seek a single metric to evaluate the model's performance.

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

2 Softmax Regression

In softmax regression, the output (prediction) is represented as a probability distribution over multiple classes.

It involves multiple linear functions, one for each class, and applies the softmax function to convert the raw outputs into probabilities.

The softmax function ensures that the predicted probabilities sum up to one, making it suitable for multi-class classification.

Formula

$$e(z)_i = \frac{e^{z_i}}{\sum_{i=1}^{k} \sum e_{z_j}}$$

$e(z)$ *softmax function.* z *input vector.* e^{z_i} *standard exponential function for input vector.* k *number of classes the multi-class classifier.* e^{z_i} *standard exponential function for output vector.*

Loss Function

Softmax regression typically uses the cross-entropy loss function, also known as the negative log-likelihood loss, to measure the difference between predicted probabilities and true class labels.

The loss function is minimised during training to improve the model's ability to correctly classify instances into their respective classes.

During prediction, softmax regression computes the probabilities for each class using the learned parameters and selects the class with the highest probability as the predicted class for a given input.

Formula

$$\text{Cross-Entropy Loss} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{k} {y_k}^i * \log(\hat{{y}}_k{}^i)$$

m *represents the total number of samples in the dataset.* k *represents the total number of classes in the dataset.* ${y_k}^i$ *is the indicator function that indicates whether the* $i-th$ *sample belongs to class* k*. It equals* 1 *if the sample belongs to class* k*, and* 0 *otherwise.*

3 Polynomial Logistic Regression

Polynomial Logistic Regression is for non-linear decision boundaries by incorporating polynomial features.

Polynomial logistic regression extends the logistic regression model by allowing the independent variables (features) to be raised to different powers, creating polynomial terms.

Formula

$$\text{logit}(p) = \beta 0 + \beta 1 x_1 + \beta 2 {x_1}^2 + \ldots + \beta n {x_1}^n$$

Here, p represents the probability of the dependent variable being in a certain class. $x_1, 2{x_1}^2, \ldots, n{x_1}^n$ are the polynomial terms of the independent variable. $\beta 0, \beta 1, \ldots, \beta n$ are the coefficients to be estimated during model training.

Training a polynomial logistic regression model involves estimating the coefficients $\beta 0, \beta 1, \ldots, \beta n$ that best fit the training data. This can be done using optimisation techniques such as gradient descent or by solving the normal equations.

Once the model is trained and the coefficients are estimated, predictions can be made by changing the values of the independent variables into the equation. The predicted probabilities can then be converted into class labels based on a chosen threshold.

We can get idea to use precision, recall, F1 score, and confusion matrix, similar to traditional logistic regression models to which threshold is good.

4. Geometric

Geometric is one another approach for logistic regression.

Hyperplane

$w^T x + b = 0$

$w^T x$ represents the dot product of the weight vector w and the feature vector x. This part represents the linear combination of features weighted by the coefficients. b is the bias term, which can be thought of as an offset that allows shifting the hyperplane away from the origin. It controls the position of the hyperplane in the feature space. From a geometric perspective, this formula represents a hyperplane in a high-dimensional space.

Distance (di) of any point xi from plane $di = \frac{w^T xi + b}{\| w \|}$

When w and xi are in the same direction i.e. $w^T xi > 0$, then $y = 1$ Similarly, when w and xj pointing in the opposite direction i.e. $w^T xj < 0$ then $y = -1$

If $yi = 1$ so it is positive point and $yi = -1$ so it is negative point.

If $w^T xi > 0$ classifier indicating its a positive point and $w^T xj < 0$ classifier indicating its a negative point.

Look at some cases

Case 1

If positive point and classifier indicating its a positive point $yiw^Txi>0$ *which means the plane is correctly classifying the point.*

Case 2

If negative point and classifier indicating its a negative point $yiw^Txi>0$ *which means the plane is correctly classifying the point.*

Case 3

If positive point and classifier indicating its a negative point $yiw^Txi<0$ *which means the plane is incorrectly classifying the point.*

Case 4

If negative point and classifier indicating its a positive point $yiw^Txi<0$ *which means the plane is incorrectly classifying the point.*

This is what we need to achieve and in order to do that, we need to find the optimal w, which will solve this maximisation problem, as both y and x are fixed. So this is a mathematical problem which we have achieved that we would term as 'Optimisation Problem'.

$$w = \text{argmax}(\sum_{i=1}^{k} yi\, w^T xi)$$

There are several hyperplanes, and for each plane, there is a unique 'w*'. So we need to find the optimal '*w*'. The maximum value and '*w*' would give us our best plane which would be our decision surface.*

Here one problem arrives with this formula is it can impacted by outliers. In some cases, even a single outlier can have a large impact and can result in the model performing badly.

Now, we will applying sigmoid function.

$$w = \text{argmax}(\sum_{i=1}^{k} \frac{1}{1 + e^{-yiw^T xi}})$$

Above we have discussed about sigmoid function.

Why sigmoid function ?

1. *It is easily differentiable.*

2. *This property helps in reduce the impact of outliers in the dataset.*

Even for further transformation to optimise equation we can use log function.

Why log function ?

1. *Numerical Stability*

2. *Mathematical Convenience*

Issues with this optimisation problem

Regularisation helps to prevent overfitting or underfitting by discouraging the model from learning overly complex patterns in the training data. It can improve the generalisation performance of the model on unseen data. The choice between L1 and L2 regularisation depends on the specific problem and the desired properties of the model.