**Logistic Regression for handling multiple classes (k-classes) | Multinomial Logistic Regression**

A paper with text and a box

Description automatically generated

A paper with text and symbols

Description automatically generated

For this example: if we have three classes: Green, Blue and Red When we will work on the Green, we will use the Green as 1 and the rest of the classes as zeros. Again, when we will work on the Blue, the element of the Blue will be one, and the rest of the classes will be zeros.

A number with black text

Description automatically generated

**Important equations and how it works:**

Logistic regression uses a sigmoid function to predict the output. The sigmoid function returns a value from 0 to 1. Generally, we take a threshold such as 0.5. If the sigmoid function returns a value greater than or equal to 0.5, we take it as 1, and if the sigmoid function returns a value less than 0.5, we take it as 0.

A black numbers and a plus

Description automatically generated

**z** is the input features multiplied by a randomly initialized value denoted as theta.



Here, **X** is the input feature. In most cases, there are several input features. So, this formula becomes big:



**X1, X2, X3** are input features and one theta will be randomly initialized for each input feature. Theta0 in the beginning is the bias term.

**Cost Function and Gradient Descent**

The cost function gives the idea that how far is our prediction from the original output. Here is the formula for that:



Here,

* **m** is the number of training examples or the number of training data,
* **y** is the original output label,
* **h** is the hypothesis or the predicted output.

This is the equation for the gradient descent. Using this formula, we will update the theta values in each iteration:



While alpha is the learning rate sat by the user (e.g. [0.01, 0.001, 0.0001])

The optimization process involves finding the model parameters (θ) that minimize this cross-entropy cost function, typically using optimization algorithms like Gradient Descent, Mini-Batch Gradient Descent, or more advanced methods like Adam, which we discussed earlier in the context of other algorithms.

The concepts of Precision, Recall, F1 Score, Accuracy, and the Confusion Matrix in the context of binary classification:

**Precision:**

Precision is a performance metric that measures the proportion of correctly predicted positive instances (true positives) out of all instances predicted as positive (true positives + false positives). It is used to evaluate the precision of positive predictions made by a model.

Formula for Precision:

​A high precision value indicates that when the model predicts a positive class, it is likely to be correct. However, high precision may come at the cost of missing some positive instances.

**Recall (Sensitivity or True Positive Rate):**

Recall is a performance metric that measures the proportion of correctly predicted positive instances (true positives) out of all actual positive instances (true positives + false negatives). It is used to evaluate the ability of the model to correctly identify positive instances.

Formula for Recall:​

A high recall value indicates that the model is good at capturing positive instances, but it may also produce false positive predictions.

**F1 Score:**

The F1 Score is the harmonic mean of precision and recall. It is a single value that balances both precision and recall. The F1 Score is used when there is a need to consider both false positives and false negatives as equally important.

Formula for F1 Score:

The F1 Score is useful when there is an imbalance between positive and negative classes, and it provides a more balanced measure of model performance.

**Accuracy:**

Accuracy is a performance metric that measures the proportion of correctly predicted instances (both true positives and true negatives) out of all instances. It is a general measure of how well the model performs overall.

Formula for Accuracy: ​

​While accuracy is a commonly used metric, it may not be suitable for imbalanced datasets where one class is significantly more frequent than the other.

**Confusion Matrix:**

A confusion matrix is a table used to evaluate the performance of a classification model. It summarizes the predictions made by the model and compares them to the actual class labels.

|  |  |  |
| --- | --- | --- |
|  | **Predicted Negative** | **Predicted Positive** |
| **Actual Negative** | True Negative (TN) | False Positive (FP) |
| **Actual Positive** | False Negative (FN) | True Positive (TP) |

* True Positive (TP): The number of instances of the positive class correctly predicted as positive.
* True Negative (TN): The number of instances of the negative class correctly predicted as negative.
* False Positive (FP): The number of instances of the negative class incorrectly predicted as positive.
* False Negative (FN): The number of instances of the positive class incorrectly predicted as negative.

The confusion matrix allows us to calculate metrics such as Precision, Recall, F1 Score, and Accuracy using the formulas mentioned earlier. It provides a more detailed view of the model's performance than just the overall accuracy.