**ENGR 518 PROJECT REPORT School of Engineering**

**Faculty of Applied Science**

**University of British Columbia**

**Project Title:** Circle Detection

**Group No.:** 01

**Members:** Almir, Dharshini, Sayar

**Date:** December 14th, 2021

# **1. Introduction**

A manufacturer wants to build a system which detects and locates the ball in a soccer field in order to help referees determine if there is an offside/goal in uncertain situations as shown in Figure 1. The manufacturer wants to use a camera-based system which checks if an image has a  
circular shape in it or not, after which it adds additional processing.

The Circle Detection project aims to develop a machine learning application according to the aforementioned requirements. Essentially, the model must classify images based on the presence of a circle in them. As in most cases, the only circle in these images taken by the camera shall be of the football, the model will therefore predict the presence of the ball in the image as a simple ‘Yes’ or ‘No’. To be sure that any eventual circle will be detected, the algorithm will segment the new image into smaller images and classify each segment to check if it has a circle or not. Detection of a circle in a segment shall imply the presence of a circle.  
This report provides insight regarding the developmental features behind the Machine Learning Model for Circle Detection, its underlying theory, prediction accuracy, shortcomings and probable ways & methods to improve.

Bola de futebol na grama

Descrição gerada automaticamente com confiança média

Figure 1

**2. Theory**

As per the requirements of our Project, an image can either contain or not-contain a circle. For successful determination, the developed model must therefore classify images successfully, which will depend upon what the machine learns. We call a supervised machine learning problem two-class (or binary) classification when the output of a dataset has just two discrete values, commonly referred to as two classes. Two-class classification is a type of regression in which the data is presented as P input/output pairs {() with each input being an *N* dimensional vector. The matching output , on the other hand, is no longer continuous and only accepts two discrete numbers. The Classification problem may be formulated in different ways, through different perspectives (like, Logistic Regression, Perceptron, and Support Vector Machines. While these perspectives widely differ on the surface, they all reduce to virtually the same essential principle for two-class classification). [1] In our case of Circle Detection, we have used Logistic Regression and employed the Cross-Entropy Cost Function.

In Logistic Regression, we'll assume that our data's output has a value of 0 or +1, i.e., is 0 (Circle) or +1 (No Circle). The output values are frequently referred to as labels in the context of classification, and all points with the same label value are referred to as a data class.

Essentially, a step function could be employed to describe the regression boundary and a corresponding Least Squares cost function could be generated for further tuning the weights, but unfortunately it is very difficult (if not impossible) to properly minimize this Least Squares cost using local optimization, as at virtually every point the function is completely flat locally. This problem, which is inherited from our use of the step function, renders both gradient descent and

Newton’s method ineffective, since both methods immediately halt at flat areas of a cost function. Hence, we use a continuous approximation that matches the step function closely, called the logistic sigmoid function (Equation 1):

σ (x) = …(Eq. 1)

Employing the sigmoid in place of the step function and finding the point-wise cost as a log error, we arrive at the Cross-Entropy Cost which essentially penalizes violations of our desired  
(approximate) equalities, much more harshly than a squared error does. This Cross-Entropy Cost function (formed by taking the average of the point-wise costs over all P data points) can be written as (Equation 2):

= - … (Eq. 2)

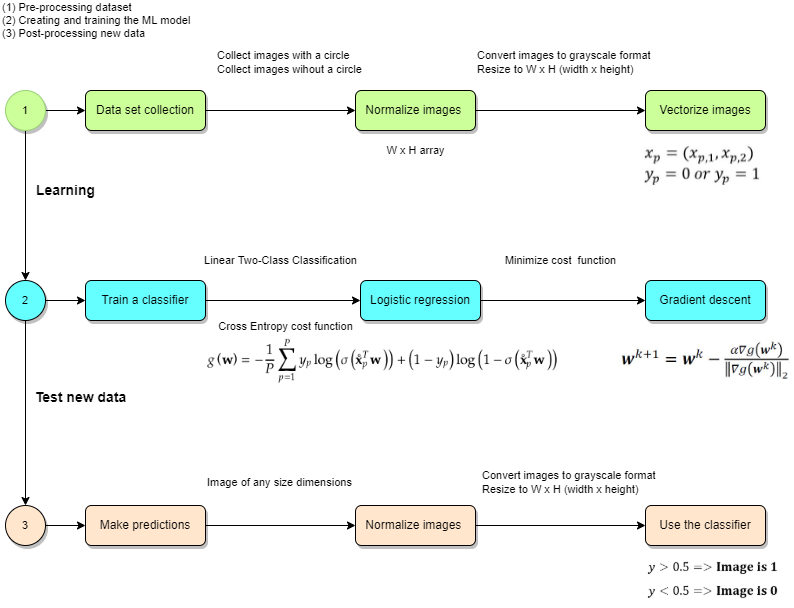
To minimize this cost we can use any local optimization method. In our model we have computed the gradient (as shown in Equation 3) of the Cross Entropy cost in closed form and applied it in the Gradient Descent method (as shown in Equation 4) for minimizing the cost. [1]

… (Eq. 3)

… (Eq. 4)

**3. Algorithm**

We have used Python Language to write the algorithm and produce our model. The algorithm can be divided into 3 main steps; namely Creating & Pre-processing the Dataset, Creating & Training the ML Model & Post-processing test data.

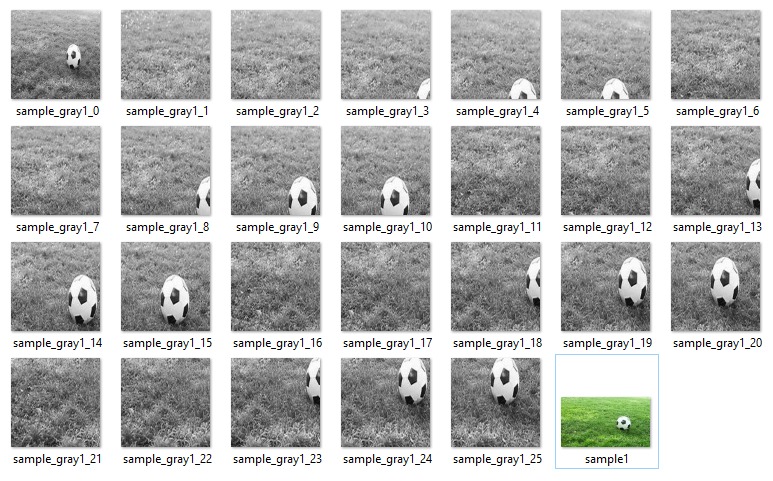
****

**Creating & Pre-processing the Dataset**: Primarily, we obtained the dataset of images of soccer balls taken through a robot camera from the IEEE DATAPORT: Open Soccer Ball Dataset. [2] The images without a soccer ball or circle were manually imported. In total, we collected 300 images with circles and 100 images without circles for training our model. Further, these images were normalized, i.e., converted to Greyscale colour and resized into a Width X Height (W x H) array of dimensions 64 X 64 pixels each. All these images were thereafter metricized into a *numpy* array ‘*XT*’ where each image is represented as a line vector with 4096 features (64 x 64), then appended with corresponding labels with column vector ‘*Y*’ and saved as a .csv file named ‘DATASET’.

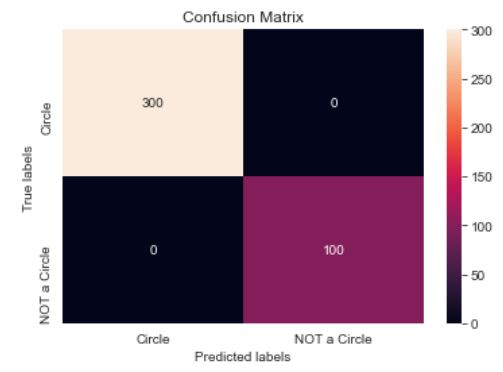
**Creating & Training the ML Model:** After finalizing the dataset, we primarily defined the ‘LogisticRegression\_Implemented’ class which further contains functions like the Sigmoid for activation and Gradient Descent for minimizing the Cost function. Thereafter, the ‘Fit\_implemented’ function had been defined, that basically generates the regression model with the following parameters: the feature array containing all the vectorized images, the vector containing the labels of each image, Learning Rate (0.00001) and Number of Iterations (2000). It should be noted that the ‘fit’ calls the functions Cross-Entropy Cost & ‘gradient descent’ (Eq.3 & Eq. 4), which are responsible for minimizing the cost in combination with the “sigmoid function” (Eq.1), by updating the coefficients through a loop in a range of the prescribed number of iterations. This process would ultimately return the optimal weights vector which gets saved in a .csv file named ‘COEFFICIENTS’. Attempts were made to include a Regularization parameter but that only increased the model’s complexity without improving the results.

Further in this part itself, 3 more functions for Fitting, Predicting and calculating the Score had been defined, which would eventually be used in testing new data.

**Post-processing test data:** Once the model got trained and the parameters that described the classification line were learned, the labels of any test data could be estimated if it were represented in a similar way (gray scaled numerical array). We have provided a ‘Test Space’ of 10 images (not part of the training dataset, although they look similar) at once in our model. Further, as mentioned before, the algorithm segments the test images into smaller images, runs the complete image to detect a circle and if no circle is detected, runs through each segment to check if it has a circle or not (for simplicity, it was assumed that whole circle stays inside the image, and its radius is around 25% of the width of the image. Shown in Figure 2). The segments were made to be overlapping to avoid missing circles which lie between segments. The results of each image gets published as a separate .csv file named ‘RESULTS\_*number*’.

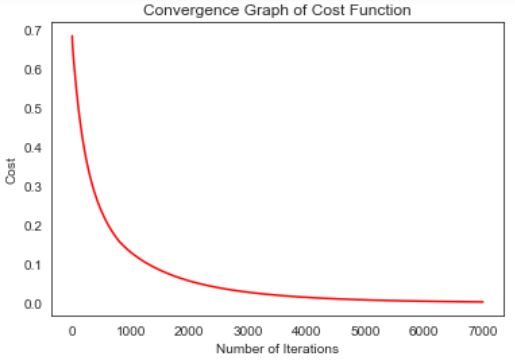
Fig. 2: Illustration of Test Image ‘sample1’ being segmented and classified

**4. Result Analysis and discussion**



After looking at the Confusion Matrix heat map, we can clearly see that our model has performed fairly well with zero outliers in total (Model Score of 1.0). Further, as we test the model with 10 different test images, fairly simple samples get rightly classified barring a few cases. This performance is mainly due to the type of dataset that has been used for training the model. We used fairly simple images of a football in action for training ‘Class 1’. Further, for training ‘Class 0’, we didn’t use just any random image without a circle. This class was trained with 100 exclusive images of empty football grounds from different camera angles. The results of the Test set have been tabulated a follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Samples** | **Dataset Label** | **Model Prediction** | **Model Error** |
| Sample1 | 0 | Circle Detected | Nil |
| Sample2 | 0 | Circle Detected | Nil |
| Sample3 | 0 | Circle Detected | Nil |
| Sample4 | 0 | Circle Detected | Nil |
| Sample5 | 0 | Circle Detected | Nil |
| Sample6 | 1 | No Circle Detected | Nil |
| Sample7 | 1 | No Circle Detected | Nil |
| Sample8 | 1 | No Circle Detected | Nil |
| Sample9 | 1 | No Circle Detected | Nil |
| Sample10 | 1 | No Circle Detected | Nil |



Thus, the Tested Dataset containing 10 images were identified accurately.

As it’s evident from the Cost Function Convergence graph, the curve flattens well at 7000 iterations and classifies the training set with zero outliers. Thus, it can be inferred that our dataset is fairly clean for the model to draw up a linear classification boundary.

The major observations that we have noted with the dataset and the circle detecting algorithm thus far are as follows:

1. The shape of the head of various players being circular shape confuses the model considerably, and such images containing players or other similar circular elements should be avoided for training and testing images.
2. Complicated images (such as images containing 2 balls or more than one circles) should be avoided for testing, as the model will be unable to distinguish in these cases.
3. Near-circular shapes like high-degree polygons can also confuse the model.

As we are using pixels as features for identifying a circular shape, due to the difference in the relative size, position in the image, and the contrast of the image itself, these features can be misleading in certain cases. The pixel values influence the model’s performance rather than the shape of objects in the image. [1]

Thus, a way to improve the model would be to use the normalized histogram of each image’s edge content instead of its pixels as features. The normalized histogram (unlike raw pixel values) would then represent an image grossly while ignoring the location and ordering and could potentially improve the model’s performance along with having to extract less processing information, as by taking edges instead of pixel values we significantly reduce the amount of information we must deal with in an image without destroying its identifying structures. [1]

**5. Conclusion**

The results of our Machine Learning Model shows that it is able to detect a circle in an image with a fair degree of accuracy. So, if the manufacturer provides images from various areas of the football field, the model would provide satisfactory results. Further testing of the model with different kind of images/ from different camera angles would reinforce the accuracy. As mentioned before, it is recommended that the manufacturer sticks to images with less number of unnecessary elements in the images for testing.

For further improvements:

* The training dataset could be increased with suitable images
* The ML Algorithm in its totality could be modified to use Edge Histogram as features.

**6. References**

1. Jeremy Watt, Reza Borhani, and Aggelos K. Katsaggelos. 2016. “Machine Learning Refined: Foundations, Algorithms, and Applications (1st. ed.)”. Cambridge University Press, USA.

2. Reinaldo Augusto da Costa Bianchi, Danilo Hernani perico, Thiago Pedro Donadon Homem, Isaac Jesus da Silva, Douglas de Rizzo Meneghetti , September 21, 2020, "Open Soccer Ball Dataset", IEEE Dataport, doi: <https://dx.doi.org/10.21227/0vvr-5c61>