Project:

**Implementing Regularized Logistic Regression with K-fold cross validation and RBF, optimizing using Gradient Descent in Synthetic Data.**

- first show linear basis function (without RBF) plots and the prediction accuracies. Then show the RBF prediction accuracies. Discuss about them

- discuss about what we used such as Logistic regression, K-fold cross validation, Radial basis functions and Gradient descent

- generate output and take screenshots as necessary for error comparison

- talk about improvement ideas and problem faced....

A two-class problem was chosen from the <https://www.kaggle.com/> real world data-repository.

**Implementing Regularized Logistic Regression with K-fold cross validation and RBF, optimizing using Gradient Descent in real image data.**

- As image data are multi-dimensional one can never be sure if it will be linearly separable or not. So we used RBF so that it can take care of the non-linearity.

- The original RGB images had a Design matrix of dimensions 2800 x 19200. We experienced the curse of dimensionality implement RBF in these high dimension feature vectors. It was time consuming.

- To reduce the dimensionality we transformed the RGB image into (i) grayscale images and (ii) B&W edge image. Even after that the Design matrix was 2800 x 6400. To obtain further reduction of dimension while keeping the important features intact, the image was shrunk by 25%. The Design matrix dimensions became 2800 x 400 which worked well.

**Introduction:**

- talk about the dataset here... and the pre-processing. I have copied some texts from the kaggle link: <https://www.kaggle.com/rhammell/ships-in-satellite-imagery>

The dataset consists of image chips extracted from Planet satellite imagery collected over the San Franciso Bay area. It includes 2800 80x80 RGB images labeled with either a "ship" or "no-ship" classification.

Provided is a zipped directory shipsnet.7z that contains the entire dataset as .png image chips. Each individual image filename follows a specific format: {label} \_\_ {scene id} \_\_ {longitude} \_ {latitude}.png

label: Valued 1 or 0, representing the "ship" class and "no-ship" class, respectively.

scene id: The unique identifier of the PlanetScope visual scene the image chip was extracted from. The scene id can be used with the Planet API to discover and download the entire scene.

longitude\_latitude: The longitude and latitude coordinates of the image center point, with values separated by a single underscore.

The dataset is also distributed as a JSON formatted text file shipsnet.json. The loaded object contains data, label, scene\_ids, and location lists.

The pixel value data for each 80x80 RGB image is stored as a list of 19200 integers within the data list. The first 6400 entries contain the red channel values, the next 6400 the green, and the final 6400 the blue. The image is stored in row-major order, so that the first 80 entries of the array are the red channel values of the first row of the image.

The list values at index i in labels, scene\_ids, and locations each correspond to the i-th image in the data list.

**Class Labels**

The "ship" class includes 700 images. Images in this class are near-centered on the body of a single ship. Ships of different ship sizes, orientations, and atmospheric collection conditions are included. Example images from this class are shown below.



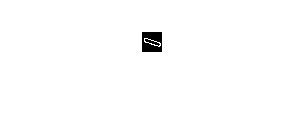
The "no-ship" class includes 2100 images. A third of these are a random sampling of different landcover features - water, vegetion, bare earth, buildings, etc. - that do not include any portion of an ship. The next third are "partial ships" that contain only a portion of an ship, but not enough to meet the full definition of the "ship" class. The last third are images that have previously been mislabeled by machine learning models, typically caused by bright pixels or strong linear features. Example images from this class are shown below.



Acknowledgements

Satellite imagery used to build this dataset is made available through Planet's Open California dataset, which is openly licensed. As such, this dataset is also available under the same CC-BY-SA license. Users can sign up for a free Planet account to search, view, and download their imagery and gain access to their API.





- talk about /show the convergence plot and discuss

- talk about /show the error plot and discuss. what do we expect in RBF compared to the linear?

- talk about /show the prediction average error and time comparison.

- I think using linear basis function and radial in grayscale image would suffice ; we do not need to include edge image; but as bonus work we can. But I do not see anything fruitful there.

- we have to take average of the time estimates so it's better to create a table and do multiple runs of the gray scale code ( linear / RBF) and record the time observed in a table and find average.

- talk about improvement ideas and problem faced....