Estimating Vehicle Locations From Satellite Imagery via a K Nearest Neighbor Classification Machine Learning Algorithm

Morgan E. Urdaneta

***Abstract*— Satellite systems such as Global Positioning System (GPS) used for radionavigation have revolutionized our modern society with a plethora of military and non-military applications. For decades, one specific type of satellites, denominated Earth observation satellites has provided high quality imagery of our planet, transforming our understanding of meteorology, geology, and cartography. Satellite imagery of a city provides important information such as the number of vehicles and the particular distribution of these throughout a specific area. This data is commercially and logistically relevant. For instance, determining the amount and density of vehicles could predict less-congested routes, as well as, demographics statistics. Nevertheless, due to the size of satellite imagery databases, processing these images by hand is an extremely time-consuming task. Here, we apply a machine-learning classification algorithm called K-Nearest Neighbor to identify red vehicles on satellite imagery. A training dataset was used to extract specific features of the vehicles to train a model (via cross-validation) and make predictions on non-labelled test imagery. After several iterations defining optimal parameters, the algorithm reliably predicted the location of red vehicles on test satellite image.**

***Index Terms*— Satellite Imagery, Vehicle Detection, Machine Learning, K Nearest Neighbors, Cross-Validation**

I. Introduction

SINCE the advent of satellite technology at the end of the 1950's, the way we analyze with our surroundings has drastically changed [1]. Data from Global positioning systems (GPS) and satellite imagery have found a myriad of military and non-military applications. Satellite imagery gave a leap forward impulsed by the military potential of spy satellites as an espionage and reconnaissance tool [1]. However, current applications of this type of data widely extend this purpose. Satellite imagery has revolutionized the many fields of study including meteorology, cartography, and geology [1]. Moreover, object classification of this type of high-resolution data over a wide area has a great commercial, geopolitical, and military interest. For instance, vehicle detection in a region of interest could provide data on less-congested routes, conglomerated zones and escape routes in case of crises, and enemy positions in the commercial, geopolitical, and military settings, respectively. However, due to the size of these imagery databases, manually labeling objects of interest could be a very time-consuming task. In order to overcome this issue, machine learning algorithms could be implemented to predict the location of these objects [2]. Machine learning (ML) uses statistical methods to allow computers to progressively predict outcomes without being explicitly programmed [2]. One MLL method commonly used in classification is the K-Nearest Neighbor (KNN) algorithm [3]. KNN classifies objects based on how close a point *k* lies within the training feature space [3]. Here, to identify red vehicles on satellite imagery dataset. A training dataset was used to extract specific features of the vehicles to train and test the accuracy of different models (via cross-validation) and make predictions on non-labelled test imagery. After several iterations defining optimal parameters and appropriate methodology, our ML algorithm reliably predicted the location of red vehicles on a sample region of interest of testing satellite imagery.

II. Implementation

The implementation of our machine learning algorithm to identify the location of red cars in satellite imagery consisted on a series of systematic phases for training and testing data. For the training data, the initial step was to collect, and label data with specific features of interest to achieve our goal. From our training satellite imagery (*data_train.npy*) and a small dataset (n=28) of labelled red vehicles (*ground_truth*), we decided to extract features based on color. In order to visualize the color profiles from this set of labelled cars, we plotted the RGB profiles in 3D, identifying each point with its respective color (Fig 1A). Since the training data was relatively small for our classification goal [3], we decided to extend our training data with RGB features of more red cars. By incrementing our labeled training data sample size, we had a more accurate representation of our features. Nevertheless, as the sample size increased the number of potential outliers increased as well. These outliers could deter the performance of our machine learning classifier [2]. These labelled outliers could've been obtained by the dark shadows (dark points) or sun reflection (clear points) from the rooftops of the bigger data set (Fig 1B).

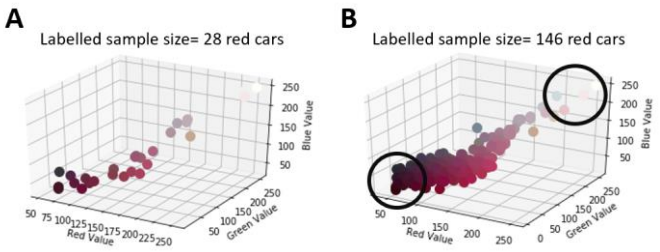

Figure 1. RGB profiles of the *ground_truth.npy* training labelled data set. RGB profiles of additional training labelled data. The black circles represent potential outliers from Class 1.

The next step in our ML pipeline was to select a model to classify our extracted features. We decided to use a KNN [3] or K Nearest Centroid [4] algorithm for this task (See Experiments section). Unfortunately, due to the nature of this algorithm, our intended class needed to be contrasted with a second class whose features could be easily discerned via Euclidean distance. Namely, we needed to label RGB values of non-red cars. Once we gathered more labelled data from different colors as well as random samples within the training dataset, we ran the KNN algorithm and tested its performance to classify RGB training features via K-fold (K=30) cross-validation. In parallel, we normalized the RGB values to visualize the performance of our classifier in two dimensions (See Experiments section). After the K-fold Cross-validation offered optimal number for neighbors, we performed the classification on a small sample of the testing data. This testing data corresponded to the RGB values of a randomly selected small sample of pixels within the testing data imagery (*data_test.npy*). The results of the classifier consisted on the class labels to which the given testing pixel RGB values belonged (i.e. Class 1: Red cars, Class 2: Non-red cars). Finally, to visualize the effectiveness of our classifier, we indexed each label corresponding to class 1 (red cars) to obtain the RGB values of each element. Then, we applied the built-in function *np.where* to obtain and mark the coordinates of the pixel with that RGB profile.

III. Experiments

In order to properly identify red vehicles within our test data, we had to perform a series of experiments in our training, validation data sets. Coincidentally, the order these series of experiment corresponds to a common approach of the machine learning pipeline [2].

A. Labelling Training Data and Extracting features

The first step in our machine learning methodology was to label training data for our algorithm. Hence, we used the pixel locations of labelled red vehicles (*ground_truth*) from our training satellite imagery (*data_train.npy*). Since the goal of our algorithm was to find red cars, we had two potential features to extract from the feature data: car dimensions, and color. Due to the variability in car size and morphology, potentially different image resolution between data sets, and the author's inexperience with coding, extracting car features based on size was discarded. On the other hand, by looking at the training data set, we observed substantial color saliency [6] of red vehicles compared to other objects or pixels in the image. Hence, we decided to base our feature extraction on color [6]. To achieve this, the pixel car locations were converted into its corresponding RGB values and plotted with the corresponding RGB profile for each scattered point (Fig 2A).

B. Increasing labelled training data sample size and dealing with noise

To overcome the low sample size of the ground truth data set [2], we increased our labelled data with more cars and grabbed the pixels surrounding the car's roof to characterize their RGB values. After increasing the sample size of our labelled data, we plotted it and observed that as the sample size increased, the noise of our features increased as well [2] (Fig 2B). The way this noise was represented in our data set was through very dark or very clear shades of red. This noise could significantly decrease the performance of our classifier [3]. From our plots, we observed that a big portion of our data was discretely clustered in the RGB space, this gave us clues regarding what machine learning algorithm was optimal for our intended goal. In addition, we could overcome the noise issue by labelling several data points with RGB profiles similar to the noise and classifying them as Class 2 (other cars). Using this approach, we could mask these outliers. In the case of a KNN algorithm [3], a *k* test point that is close to a very clear tone of red would be classified as Class 2-(other cars) instead of Class 1 by adjusting the n of neighbors to an optimal value.

C. Improving classification errors – more labelled training data.

Post-addition of the class 2 labels for black and white outliers, we briefly applied a KNN non-parametric classifier. However, anecdotal evidence from the testing output showed that the predictions were unpredictable and highly variable, with a high rate of false positives. For instance, the classifier was classifying some color points (i.e. teal, green, blue) as red. One possible cause of this issue was that our classified data was just a diagonal cluster in the RGB space, in which colors such as blue and yellow were neighboring at a relatively similar distance from class 2 than to class 1 (See circled clusters in Fig 3B). Hence, we obtained more labelled training data from equidistant points on the RGB feature space as well as sample RGB values from random pixels within the training dataset (Fig 2C)

D. Fitting and testing 2D Normalized model.

As previously stated, visualizing our RGB feature space contributed to potential solutions such as classifying new labelled training data near the outliers. We attempted to reproduce this visualization approach of the KNN with a normalized RGB color space (1) to obtain the r-b 2D spread [2]

$$normalized\ R = \frac{R}{R+G+B} \quad (1)$$

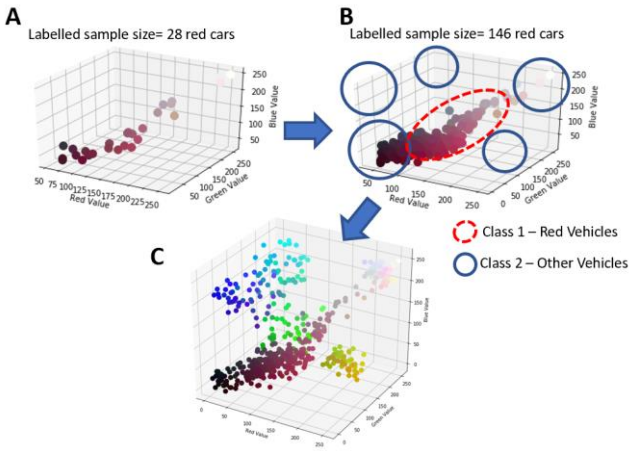

Figure 2. Pipeline of color feature extraction for training data for the KNN algorithm. A) Low sample size labelled training data. B) Addition of 118 labelled red cars from the training satellite imagery. Red dashed circle represents Class 1 while blue circles represent empty spaces in the RGB space representing potential colors for other vehicles (Class 2). C) Addition of red Class 2 labelled values for training purposes.

This normalization scheme allowed us to visualize the classification boundaries of our algorithm even with a 3-class classification scheme (Fig 3). However, reverting the normalized values into pixel locations for testing data was challenging. Additionally, there are problems associated with nearest neighbors and changes in dimensionalities. As stated by Beyer *et al*, adjusting the number of dimensions of the feature space, in complex data (like images), the contrast in distances of neighboring points also changes [3].

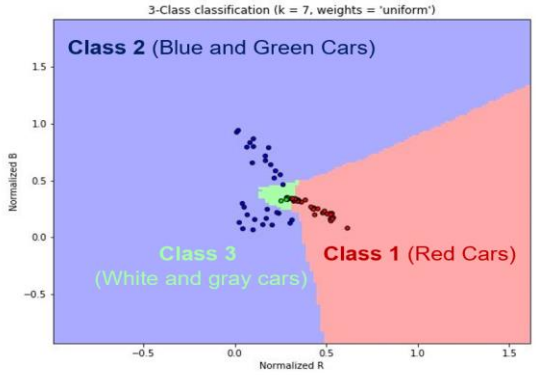

Figure 3. Visualization of the KNN 3 class classification approach with a normalized RGB color space (r-b 2D spread)

E. Choosing model through training and validation data.

After reverting our feature space to the standard RGB space, we split our labelled training data (Fig. 2C) in valid and training data. We then proceeded to systematically test via K-fold cross-validation [7] different models and parameters [8]. We used a uniform KNN algorithm and a KNN with dissimilarity measure based on Euclidean distance (2) [4]. Additionally, we tested a K nearest centroid model [5] based on the assumption that the mean of the red-car cluster would predict the correct classification label.

$$d_E = \sqrt{(\mathbf{x}_1 - \mathbf{x}_2)^T (\mathbf{x}_1 - \mathbf{x}_2)} \quad (2)$$

For each KNN classifier, we repeated our cross-validation experiments with low, medium and high number of neighbors. Analogously, we used a low, medium and high shrink threshold for the nearest centroid algorithm (Fig. 4).

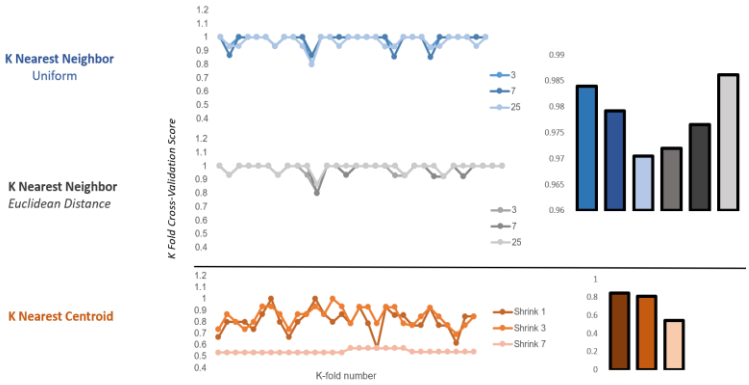

Figure 4. Visual representation of a 30 K-fold cross-validation of Uniform and Euclidean distance KNN and K Nearest Centroid classifiers. Each model was tested with arbitrary low, medium and high number of neighbors of shrink thresholds. The Color-coded bar graphs represent the mean accuracy of each experimental group.

From the results of figure 5, we can get into the conclusion that uniform and Euclidian's distance KNN's offers the best results, especially when the number of neighbors remains constant. We proceeded with a Euclidian's distance KNN with a 7 number of neighbors. One tool to visualize the performance of classifiers through a type of contingency table denominated the confusion matrix [2]. In the given case that our algorithm must classify between different classes of cars, this tool would allow us to obtain a detailed description of the classifier's performance with respect to each class [2] (Fig. 5).

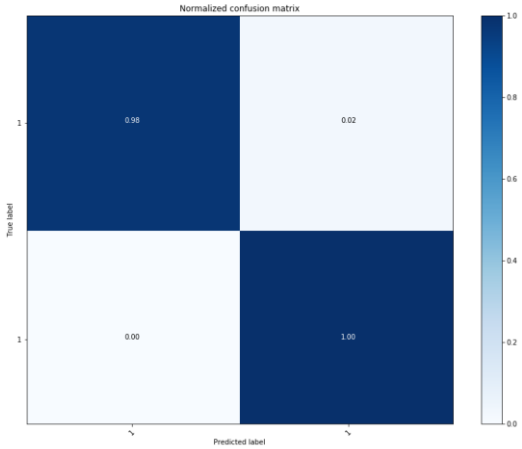

Figure 5. Confusion matrix of the KNN algorithm whose dissimilarity measurement was set to Euclidian's distance (2) and a total of 7 neighbors.

F. Testing the performance of the KNN classifier on non-trained data

After the KNN classifier predicted the classes of the testing data input (RGB values of each pixel), the indexes labeled class 1 (red cars) were used to extract the corresponding RGB values of each element. As previously done with the training data, the *np.where* function was used to identify (and label) the corresponding (red car) pixels in the testing satellite imagery. (Fig 6).

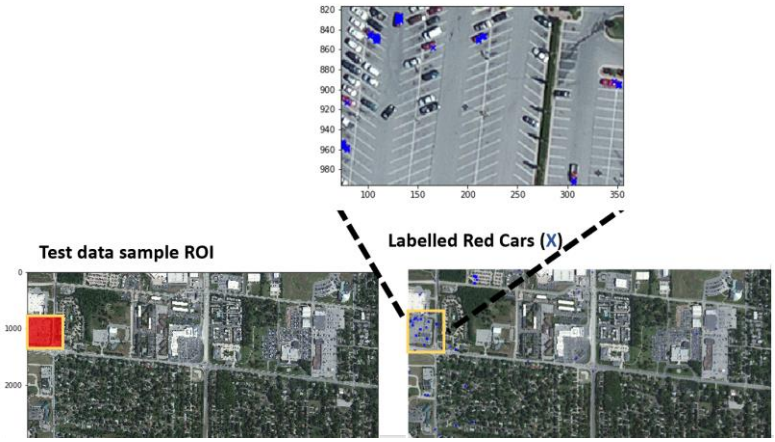

Figure 6. Testing the KNN classifier with test satellite imagery, a test data sample region of interest (ROI) was selected and the pixels that corresponded to Class 1 (Red cars) were marked with a blue x.

IV. Conclusions

Artificial intelligence and machine learning algorithms have been a transformative tool for many fields. However, the implementation of these tools might present several challenges. Throughout this project, we encountered several of these challenges. From the implementation to the experiments, this project has taught me that a great portion of the time invested in machine learning consists on organizing the data. Namely, extracting the correct features from training data, making sure that the dimensionality of such features is adequate, performing normalizations, etc. On the other hand, just a small percentage of time is spent on choosing and testing the model. Another important point that is pertinent to note from our experiments, is the inherent limitations of machine learning. Even though our KNN algorithm was successful in marking pixels of the test data based on color features, our implementation still has several drawbacks: 1) The running time of the algorithm not feasible for real time applications. 2) Basing our feature extraction solely on the RGB space of individual pixels could increase the number of false positives (i.e. a red object could be mislabeled as a vehicle). This could be improved by applying specific size conditions (i.e. more than x class 1 pixels equals one red vehicle). Nevertheless, this adjacent pixel method would fail with testing satellite data of different resolution, in which a single vehicle might be a couple of pixels or hundreds of them. Regarding this last point, as we saw with the beer foam example in class, the level of prediction of a model is a good as the training data. Extrapolating untrained features to a new test dataset will give unexpected errors, unless prior assumptions are stated [2]. Similarly, the accuracy results of training data cross-validation might not directly translate into testing data. The current dataset could be improved by combining our KNN RGB features detector with additional models for object detection like a hybrid deep convolutional neural network [9]. In conclusion, ML offers key tools to wide variety of applications in several fields. The statistical methods of se could go from identifying behavioral patterns to identifying vehicles in satellite imagery. Nevertheless, as this project attests, the accuracy and perform of this technology is highly dependent on the quality of the extracted features from the training data, as well as, the implementation of our algorithms.

References

[1] J.B. Campbell and R.H. Wynne, Introduction to remote sensing, Guilford Press, 2011.

[2] N.M. Nasrabadi, "Pattern recognition and machine learning," Journal of Electronic Imaging, vol. 16, pp. 049901, 2007.

[3] K. Beyer, J. Goldstein, R. Ramakrishnan and U. Shaft, "When is “nearest neighbor” meaningful?" in International conference on database theory, pp. 217-235, 1999.

[4] K.Q. Weinberger and L.K. Saul, "Distance metric learning for large margin nearest neighbor classification," Journal of Machine Learning Research, vol. 10, pp. 207-244, 2009.

[5] R.M. McIntyre and R.K. Blashfield, "A nearest-centroid technique for evaluating the minimum-variance clustering procedure," Multivariate Behavioral Research, vol. 15, pp. 225-238, 1980.

[6]J. Van de Weijer, T. Gevers and A.D. Bagdanov, "Boosting color saliency in image feature detection," IEEE Trans.Pattern Anal.Mach.Intell., vol. 28, pp. 150-156, 2006.

[7] I.H. Witten, E. Frank, M.A. Hall and C.J. Pal, Data Mining: Practical machine learning tools and techniques, Morgan Kaufmann, 2016.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss and V. Dubourg, "Scikit-learn: Machine learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825-2830, 2011.

[9] X. Chen, S. Xiang, C. Liu and C. Pan, "Vehicle detection in satellite images by hybrid deep convolutional neural networks," IEEE Geoscience and Remote Sensing Letters, vol. 11, pp. 1797-1801, 2014.