Cactus Identification

Mikkel Bjornson, Rory Higgins, & Mary Kate Thomerson

5/30/2021

Introduction

Assessing the state of an ecosystem can be an expensive time consuming process, often requiring the deployment of teams to visually inspect large areas. The ability to count flora within an ecosystem remotely, could greatly improve the ability to track ecosystem change. Researchers in Mexico have created the VIGIA project which aims to build a system for autonomous surveillance of protected areas (López-Jiménez et al, 2019). Images from this surveillance would be used to identify if certain vegetation is present, our analysis will look for the presence of a specific type of cactus. The purpose of this project is to determine what model method is best for image classification. We have compared four different models, support vector machine (SVM), random forest, XGBoost, and Convolutional Neural Network, as well as an weighted average of all four models.

Data Description

Feature Extraction

The data set consists of 17,500 JPEG files with an additional 4000 file test set for the Kaggle competition (López-Jiménez et al 2019). Each photo is 32x32 pixels. The EBImage package is used to read each file. Each pixel is given three values corresponding to the three primary colors. Since all images are the same size, no reshaping is necessary. The list of images is converted to an array for easier computation. The images' pixels split into primary colors are used as the classification features, making a total of 3072 features per image. This provides a total of 17,500 observations each with 3,072 features. Each feature is a value from 0 to 1 that represents the intensity of one of the primary colors for a single pixel. The response variable is the presence or absence of barrel cactus. The data set is split 60% train, 20% validation, 20% test. The best models for each method are then selected and fit on the 17,500 images with a 80% training, 20% validation split. They are then tested on the 4000 image Kaggle test set.

Feature Engineering

Model fitting is limited to the features available. With the goal of improving accuracy, additional features are engineered. For each primary color, the difference between each pixel and the one immediately to the right are found. In order to maintain the 32x32 structure of the arrays, the column of the pixels on the far right are given zero values. This results in an additional 3,072 features. The new features indicate the extent that the colors change between pixels. This is then done again, using the pixels immediately below a pixel, adding a row of zeros at the bottom to maintain the 32x32 structure. The idea is that large difference indicate edges of objects and will be referred to as horizontal and vertical edge features. Models are built with horizontal edge features, and both horizontal and vertical edge features.

Machine Learning Models

Support Vector Machine

The SVM model was fit using the parallelSVM package and trained on data containing 3072 features (Hastie et al 2017, James et al 2021). The edged data was not used due to complications with the model computation. The model used a radial kernel with C-classification.

Random Forest

The random forest model was fit using the ranger package (Hastie et al 2017, James et al 2021). The model consisted of 150 trees, each built from a random sample of 301 images.

XGBoost

XGboost models are fit using the xgboost package. The xgboost model used a tree based approach with a tree depth of four (Hastie et al 2017, James et al 2021). Using the xgb.train function we set the number of boosting rounds fairly high at 2000 with an eta of 0.2. However, once the boosting failed to improve the error rate for 15 rounds this was cut short and the model with the best validation error rate was kept.

Convolutional Neural Network - Keras

A Convolutional Nueral Network model was fit using the Keras package (Hastie et al 2017). Once with the RGB levels and horizontal edge features then again with the addition of vertical edge features. Both models were fit using a batch size of 500 and 50 epochs.

Horizontal edge

```
## Model: "sequential"
## _________________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =========================================================================
## conv2d_2 (Conv2D)            (None, 30, 30, 32)        1760
## _________________________________________________________________________
## max_pooling2d_1 (MaxPooling2D)  (None, 15, 15, 32)     0
## _________________________________________________________________________
## conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
## _________________________________________________________________________
## max_pooling2d (MaxPooling2D)    (None, 6, 6, 64)       0
## _________________________________________________________________________
## conv2d (Conv2D)              (None, 4, 4, 64)          36928
## _________________________________________________________________________
## flatten (Flatten)            (None, 1024)              0
## _________________________________________________________________________
## dense_1 (Dense)              (None, 64)                65600
## _________________________________________________________________________
## dense (Dense)                (None, 2)                 130
## =========================================================================
## Total params: 122,914
## Trainable params: 122,914
```

```
## Non-trainable params: 0
## _________________________________________________________________________
```

Horizontal and Vertical Edge Features

```
## Model: "sequential_1"
## _________________________________________________________________________
## Layer (type)                 Output Shape              Param #
## =========================================================================
## conv2d_5 (Conv2D)            (None, 30, 30, 32)        2624
## _________________________________________________________________________
## max_pooling2d_3 (MaxPooling2D)  (None, 15, 15, 32)     0
## _________________________________________________________________________
## conv2d_4 (Conv2D)            (None, 13, 13, 64)        18496
## _________________________________________________________________________
## max_pooling2d_2 (MaxPooling2D)  (None, 6, 6, 64)       0
## _________________________________________________________________________
## conv2d_3 (Conv2D)            (None, 4, 4, 64)          36928
## _________________________________________________________________________
## flatten_1 (Flatten)          (None, 1024)              0
## _________________________________________________________________________
## dense_3 (Dense)              (None, 64)                65600
## _________________________________________________________________________
## dense_2 (Dense)              (None, 2)                 130
## =========================================================================
## Total params: 123,778
## Trainable params: 123,778
## Non-trainable params: 0
## _________________________________________________________________________
```

Averaged Model

Using the Random Forest , XGBoost, and CNN models, the probability of cactus presence is calculated for each image in the validation set. Then weighted means of the predictions from each model are calculated (Hastie et al 2017). Weights are determined by attempting all permutations of possible weights to an accuracy of 0.002 on the validation predictions. The weighting resulting in the lowest error is accepted. The resulting model for the images including horizontal edge features is:

$$Cactus\ probability = 0.734(CNN) + 0.266(XGBoost) + 0(Random\ Forest)$$

The weight mean for the images with horizontal and vertical edge features is:

$$Cactus\ probability = 0.478(CNN) + 0.488(XGBoost) + 0.034(Random\ Forest)$$

Test Results

Horizontal Edge Features

Error	SVM*	Random Forest	XGBoost	CNN	Averaged
Training	92.9%	>99.9%	>99.9%	>99.9%	
Validation	93.1%	92.7%	96.4%	97.6	98.3%
Test	63.7%	91.3%	95.6%	97.5%	97.7%

- SVM model did not contain any edge features.

Best models were fit using the SVM, Random Forest, XGBoost, CNN, and weighted averages models. The SVM model could not incorporate the Edge features and is presented here using only the RGB features of the images. The SVM model produced the lowest accuracy rate out of all the models and was excluded from further analysis. All other methods had very high training accuracy, and moderately high validation and test accuracy. The weighted average provided some gain in both validation and test accuracy above that of the individual models.

Horizontal and Vertical Edge Features

Error	Random Forest	XGBoost	CNN	Averaged
Training	>99.9%	>99.9%	>99.9%	
Validation	92.5%	96.6%	97.8%	98.0%
Test	91.0%	95.4%	97.3%	97.4%

Using the same methods, Random Forest, XGBoost, and CNN models were fit with the inclusion of the vertical edge data. This resulted in a slight loss of accuracy across all models. Without any gains in accuracy, it is likely these features did not add any meaningful information.

Kaggle Competition

XGBoost	CNN	Averaged
93.8%	95.6%	96.1%

Based on the results from the previous models, the XGBoost, CNN, and weighted average models were selected for submissions to the Kaggle competition. The models were fit without the vertical edge features. Of the two models built on a single method, the CNN resulted in higher accuracy. However, similar to results above, the weighted average resulted in an increased accuracy above either model separately.

Conclusion

The Convolutional Neural Network consistently gave higher accuracy than the other three base models regardless of the engineered features. However, weighted averages of the models consistently improved this method indicating each model has something different to add. If computation time is not a pressing issues, the weighted average appears to consistently give the best predictions, with an accuracy of 96.1% on the Kaggle data set. This can be a computationally expense process requiring fitting multiple models. The CNN is less expensive and provides an accuracy of 95.6%, slightly less than the weighted average. Based on this analysis the VIGIA would have the best chance of accurate classification by using the weighted average model with their surveillance of protected areas.

Citations

López-Jiménez, E., Vasquez-Gomez, J.I., Sanchez-Acevedo, M.A., Herrera-Lozada, J.C., Uriarte-Arcia, A.V.. (March 8, 2019). Aerial Cactus Identification. Retrieved [May 11, 2021] from https://www.kaggle.com/c/

aerial-cactus-identification/data.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2021). An introduction to statistical learning: With applications in R. Springer.

Hastie, T., Friedman, J., & Tisbshirani, R. (2017). The elements of Statistical Learning: Data Mining, Inference, and prediction. Springer.