Machine Learning Engineer Nanodegree

Capstone Proposal

December 30, 2017

Proposal

Statoil/C-CORE Iceberg classifier challenge hosted on Kaggle platform.
https://www.kaggle.com/c/statoil-iceberg-classifier-challenge

Domain Background

Drifting icebergs might be dangerous for naval activities, so people currently exploit different solutions to address this problem:

1. aerial reconnaissance
2. monitoring environment from shore to assess risks
3. icebergs detection on satellite images using machine learning methods

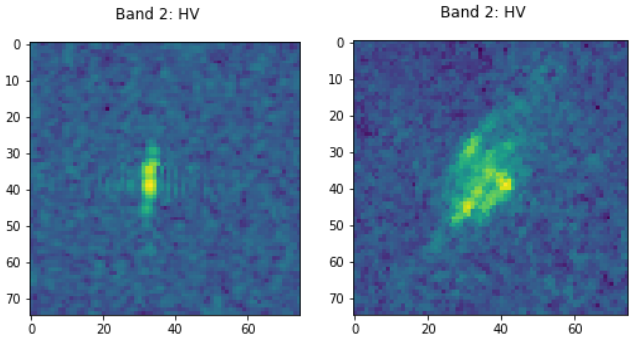
The latest approach is usually the only one available in the remote areas or areas with difficult weather conditions. The sponsor of the competition is already using one developed by C-CORE, but looking for improvements in the classification of objects in the water into two categories - ships and icebergs (binary classification).
Ship\Iceberg discrimination problem was already addressed in multiple researches. Bentes et al. [1] compared several approaches to this problem and showed that convolutional neural networks outperformed others (F1 score 98% against 88% for SVM and 84% for SVN+ PCA). Howell et al. [2] suggest quadratic discriminant (QD) with feature selection based on sequential forward selection. This approach was mentioned as the most advantageous (comparing to CNNs, unsupervised approaches like k-nearest neighbors and others) in C-Core report [3].

[1]. Ship-Iceberg Discrimination with Convolutional Neural Networks in High Resolution SAR Images http://ieeexplore.ieee.org/abstract/document/7559347
[2]. Dual Polarization Detection of Ships and Icebergs - Recent Results with ENVISAT ASAR and Data Simulations of RADARSAT-2 http://ieeexplore.ieee.org/document/4779319/
[3]. Summary of previous research in iceberg and ship detection and discrimination in SAR http://cradpdf.drdc-rddc.gc.ca/PDFS/unc196/p800002_A1b.pdf

Problem Statement

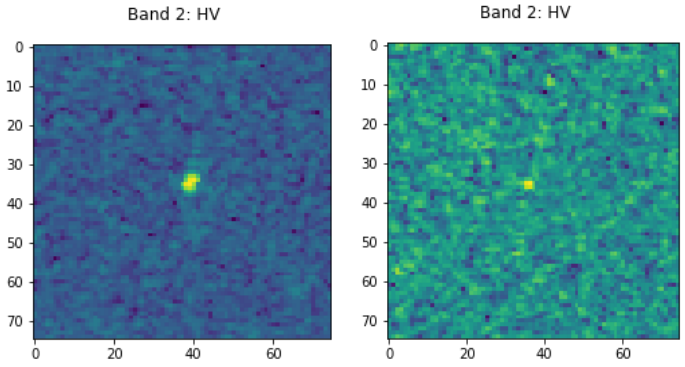
The Statoil provides images of ships and icebergs made from the space by a satellite Sentinel-1. They are centered, cropped and manually classified by human experts. The task

is to develop a model that can classify these images and achieve reasonable accuracy. As could be seen on the images below some of objects can be easily classified without any expert knowledge. For example, image (a) is clearly a ship as it has very well-defined rectangular shape. On the other hand, image (b) is an iceberg, because its shape would be very unusual for the artificially created naval object.



(a) *(b)*

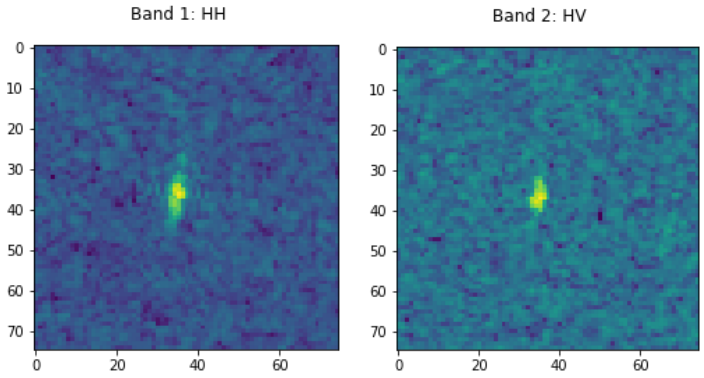
However many objects might be very challengeable for a classifier. For example, on image below objects are almost indistinguishable ((a) is a ship, (b) is an iceberg).



(a) *(b)*

Datasets and Inputs

Images are obtained by satellite Sentinel-1 using C-Band radar [1]. Frequencies used by this radar allow to make images of Earth surface even if it cannot be visually observed due to the weather conditions (clouds, fog, etc). Radar emits the signal which reflects from objects and returns back (so-called backscatter), where these echos can be recorded and translated to the image. Solid objects have stronger echoes comparing to water and appear as bright objects on the darker areas, representing water.
Sentinel-1 can emit and receive signals with horizontal and vertical polarizations, so data consists of dual-polarization images (images with 2 channels). HH channel is obtained by recording signal which was transmitted and recorded horizontally. HV channel is obtained from the signal which was transmitted horizontally, but received vertically.



The dataset contains 1604 examples and it is quite well-balanced: 753 examples of icebergs and 851 examples of ships. The dataset is organized into 2 files (training and testing data) in JSON format. Every example in dataset has following fields:

1. **id** of the example
2. **band_1, band_2** - 75x75 1-channel images in flattened form, so each is a 1D vector with 5625 numbers. Images are represented not by intensities (integers), but by strength of the backscatter signal (float numbers representing signals measured in dB units).
3. **is_iceberg** - target variable: 1 if iceberg and 0 if ship.
4. **inc_angle** - incidence angle. Means the angle, at which satellite sees the image area. This field might have “NA” values.

[1] SENTINEL-1 mission overview:
https://sentinels.copernicus.eu/web/sentinel/user-guides/sentinel-1-sar/overview

Solution Statement

The solution must be able to load dataset, preprocess data and classify it to the 2 classes - ships and icebergs. For the classifier supervised or unsupervised machine learning approaches can be used. The model should provide probabilities, that given object belongs to the particular category.

Benchmark Model

Competition sponsors do not provide any benchmark results, so I would propose several naive models to compare with:

1. Random guess model: as dataset is well-balanced and contains approximately same number of examples from both classes, random guess model showed 49% mean accuracy (on 1000 runs).
2. LogisticRegression model with 10-fold cross validation showed a mean accuracy of 65%.

Evaluation Metrics

Competition sponsor suggest to use log loss for evaluation of model performance. For binary classification logarithmic loss has following definition:

$$loss = \frac{1}{N}\sum_{i=1}^{N} y_i log\, p_i + (1 - y_i)\, log\, (1 - p_i)$$

Surely, it is already implemented by both - Keras and scikit-learn frameworks. Also, I am planning to use accuracy as more easily interpretable metric and confusion matrix as a method to identify and analyze examples, challenging for particular algorithms.

Project Design

I am planning to try following approaches to address this task:

1. Convolutional Neural Network (CNN)
2. Support Vector Machines (SVM)
3. Boosting (AdaBoosting or Gradient Boosting) with Decision Tree as a basis
4. Unsupervised approach: clustering or k-nearest neighbors

The following steps will be included in the project:

1. Data Loading: dataset is organized as files in JSON format, so it must be loaded and reorganized into 2 numpy arrays - features and targets.

2. Data Preprocessing: the images in the dataset are already cropped to the reasonable size, have square shape (75x75) and objects are approximately centered on the image. However, there are 2 images (2 bands) per sample, so they can be treated in different ways:
    a. averaged
    b. concatenated
    c. considered as 2 stacked channels (for 3D convolutions in CNN)

Depending on the algorithm, images can also be reshaped from flattened form back to the 2D shape. For example, 2D shape will be essential for convolutional neural network.

3. Validation: the dataset is very small, so k-fold cross validation will be preferable, comparing to hold-out one.

4. Data Augmentation (for CNN): again, as dataset is small, CNN likely will require aggressive data augmentation to avoid overfitting. I am planning to use built-in Keras solution - ImageDataGenerator, which allows to apply various modification such as rotations, shifts, flips, zooms, etc [2].

5. Fit\train models. For CNN, I am also planning to transfer ResNet or VGG first layers [3] into my model.

6. Optimization of hyperparameters: all mentioned algorithms have hyperparameters that needs to be optimized in order to obtain best learning results for given data. Usually, randomized approach achieves better results for search in hyperparameters space (comparing to the grid search), so I am planning to use RandomizedSearchCV implemented in scikit-learn [1].

7. Evaluation: as provided test dataset is not labeled, the predictions for it should be written to the csv format and this file should be uploded to the Kaggle platform in order to evaluate results and get loss value for the test dataset.

8. Choose best model\approach: choose model that has the smallest loss on the training dataset as a final solution.

[1]. RandomizedSearchCV
http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html
[2]. ImageDataGenerator https://keras.io/preprocessing/image/
[3]. Building powerful image classification models using very little data
https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html