kaggle Competition Project

Team Name : AAGTeam
Ahmed Amine Daou - Alejandra Jimenez - Gabriel Lagubeau

November 28, 2018

Abstract

DONT FORGET README FILE TO DESCRIBE HOW TO RUN THE CODE DOCUMENT EVERY FUNCTION,PLOTS,REMOVING OUTLIERS(it can help)

what we need to do: add data normalization(easy) + looking for missing values(easy) (won't change many things but it's important) +(maybe: removing outliers) + try to resize images to minimum possible size

1 Introduction

In this document, we'll walk through a machine learning classification problem in the context of a Kaggle competition.The purpose is to design an algorithm that can automatically identify in which class a noised hand drawn image belongs among 31 classes . The dataset is divided into a training (10 k) and a testing (10 k) sets.
your approach and results.

2 Feature Design

After loading the data from its .npy (training & testing features) and .csv (training labels) files into numpy arrays, we applied pre-processing methods to make sure our chosen machine learning algorithms work in a simpler way.

Looking for missing & infinite values :

Make sure there's no missing values in the dataset, verify if all input values are finite.

Reshaping inputs:

Reshaping training and testing images from (10000 , 2) to (10000 , 10000) . In our case, an image is a 100×100 matrix, transforming it to a vector makes things go easier.

Denoising & Centering images:

Training set images are extremely noised, and that will easily mislead the learning algorithm by giving wrong information. Our denoising algorithm conserves only the largest number of pixels connected to each other. The next step is "centering" the image, by searching the Center of mass, we translate the image by x and y to make the images superimposed which leads to more powerful learning (always changing almost the same weights) the returned result is a 50×50 image instead of 100×100, a little image containing only relevant information and faster to process.**to finish**

3 Algorithms

After data processing, we used three different algorithms:

Linear regression:

K-Nearest Neighbors:

which stores all known cases and classifies new cases based on a distance function (our case: euclidean distance). A case is classified after a majority vote of its K-neighbors.

Bagged Decision Trees:

Bagging is a method that combines the prediction from a learning algorithm like Decision Trees to make a better accurate prediction. This procedure can reduce the algorithm variance.

4 Methodology

Training/Validation split:

Our goal is to create a model that generalizes well to new data. Splitting data in a good way makes the difference. Before choosing the percentage of each set, we should be sure that we meet some conditions:
- Testing set should have the same characteristics as training set and should be big enough to give meaningful results
- Less Training data begets high variance in parameters estimation - Less Testing data begets high variance in statistical performance That's why we chose to split the testing images into (80% for training,20% for testing.

Optimization Tricks:

Image denoising:s

To get better results while training the model, firstly we began by denoising all the images using $connectedComponentsWithStats$ function from $OpenCV$ $cv2$ library which returns noiseless images based on connected pixels numbers in each image.

Denoised Image translation:

Second, To let all the images be in the same position in order to compare each other, we implemented *centerimg* function, that translates the pixels (containing relevant information(perfectly denoised image)) to the pictures top left.

Image resizing:

Third, We resized images from 100×100 to 50×50.let's test 30 x 30 have less parameters to optimize during training.

Setting hyper-parameters:

K-NN:

We testrd different k values to obtain the best accuracy. We tested k from 1 to 30. The best accuracy with the training dataset was obtained with $k = 8$.

Bagging decision trees:

Our hyper-parameters are: samples number and trees number ($n\_estimators$). By default this number is 10, but the algorithm performance is proportional to ($n\_estimators$) , but we can't choose a very hight number because it's computationally expensive. We did the test with different number of trees and we used the model with the bigger accuracy. In this case the bigger accuracy obtained was with $n\_estimators = 100$. With this model we reached our best performance.

5 Results

Algorithm	Accuracy
Simple baseline: Logistic Regression	22.85%
Best result: Bagged Decision Trees	53.16%
Second best result: K-nearest neighbors	52%

need to put some plots

6 Discussion

Discuss why your best method performs better than the simple baseline.
Which design choices where the most impactful on performance? on doit ajouter le temps d'execution de l'agorithme d'entrainement
et ici on doit lancer l'algorithme sans debruitage, apres sans recize et sans translation et comparer l accuracy et le temps d'execution a chaque fois pour dire quelle operation nous a donné plus de performance (normalement c'est le denoising qui nous a donné plus de performance)
exemple de tableau

Denoising	Image resizing	Image translation	Accuracy	Execution time
✓	✓	✓	52% +10%	sec +2 minutes

7 Statement of Contributions

We hereby state that all the work presented in this report is that of the authors