Solving Captcha Images with Machine Learning

Monika Roznere
Michael Correale

Introduction

- Problem: Develop a way to solve captcha images by extracting information from the image to determine what letters the solution is

Our Approach

- Implement Logistical Regression based on training data
  - Data has a known label
- Feed our solver data to populate class parameters
- Once it has enough data, apply the solver to predict images class
- Slowly increase the complexity as we fine-tune the solver

Tools used

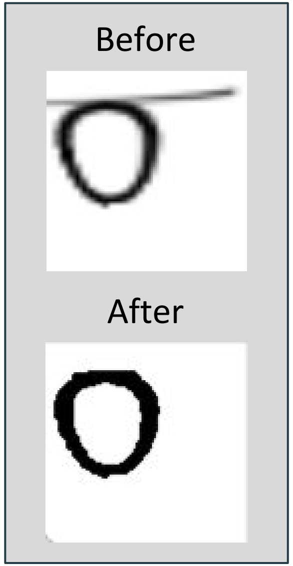
- Open CV in a C++ environment
  - Main bulk of machine learning code
- Claptcha, a simple captcha generator script written in Python
  - Generate individual characters for training
  - Later used to generate test images

Training Example

Testing Example

Dataset of Training Images

- Images of numbers / letters in the 1000s
  - Each image is the same size (100 x 100 pixels)
  - Each image has a set training label (0 to 10 for numbers)
- Each label will be trained separately
  - Using logistic regression classifiers
- 100 Images for each training label
  - Pixel values are thresholded to remove blur
  - A Closing operation is applied to reduce noise
  - Then, we save each class and a “flattened” vector of pixels



Logistic Regression

- Cost function
  - Fit the best line to our data

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log\left(h_\theta(x^{(i)})\right) - (1-y^{(i)})log\left(1-h_\theta(x^{(i)})\right)]$$

- Gradient descent
  - Want to minimize the cost function
  - Change θ to reduce J

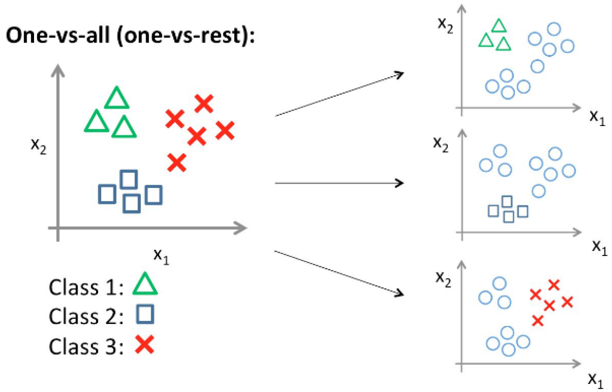
$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}((h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)})$$

- Regularized logistic regression
  - Apply Gradient Descent solution to fit the line to the data

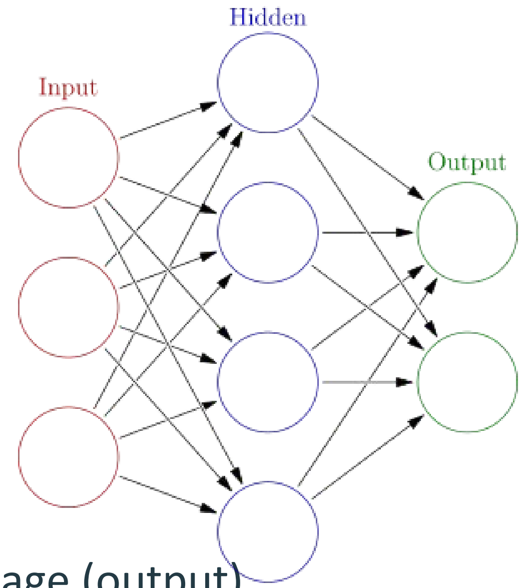
$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}[-y^{(i)}\log\left(h_\theta(x^{(i)})\right) - (1-y^{(i)})log\left(1-h_\theta(x^{(i)})\right)] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

One-vs-all Classification

- Multi-class classification
  - label each image a number from 0 to 9
- Take one class and treat the rest as “all”
  - Treat it as a separate new training set
  - Implement the binary classification
- Hypothesis for each class
- For each image input, compute probability for each class
- Chose the class with the highest probability



Neural Networks



- Multi-class logistic regression = linear classifier
    - Cannot form complex hypotheses
- Neural Network
    - Non-linear classifier; complex hypotheses
- Feedforward propagation
    - Prediction will be the label with the largest percentage (output)
- Advantages:
    - Higher percentage of accuracy
- Disadvantages
    - Decide on how many hidden nodes
    - Decide on weights for θ

General Program Flow

- Take each row of our training set
- Apply cost function with each classes parameter list
- Determine which class has the highest probability index
- Check prediction with known label
  - Adjust class parameters accordingly
- For testing:
  - Do the same process, but just take the predicted class

Thank You!
Questions?

References

- Ng, A. (n.d.). *Machine Learning*. Lecture. Retrieved December 12, 2017, from

https://www.coursera.org/learn/machine-learning/home/welcome.

- Our github link:

  https://github.com/mcorreale1/captchasolver