CS 455 Visual Information Processing

Term Project: Solving Captcha Images with Machine Learning

Monika Roznere and Michael Correale

Project Statement

For the term project, the goal is to implement an application that that can solve and identify the letters and numbers in a captcha image. In this application, we attempt to use the regularized logistic regression to train data images and apply the results to predict input captcha images. Other algorithms, such as morphological operations and simple computer vision techniques, are also implemented. Most of the features in the application are influenced by the lectures by Andrew Ng and other helpful documents of algorithm descriptions.

1 Introduction

Captcha is a prevalent technique seen on websites to test if the user on the site is an actual person and not a robot. It is a technique to prevent malicious users from sending automatic programs spam a website. In one point of view, the problem that we are solving is the algorithm that captcha applications run. A user sees a semi-deformed message, sometimes of random letters and characters, and is asked to type out the message in an input box to show that they are "real" person. Captcha applications can check the message and see if the input is actually a near close perfect match.

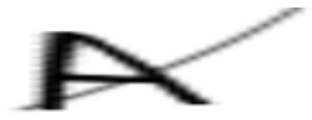
The regularized logistic regression algorithm is widely used in machine learning problems to classify and predict multiple labeled data. However, neural networks have become more prevalent for solving problems that were previously solved by regression algorithms. The main reason for the popularity of neural networks is the high percentage rate of accuracy and the ability to work with large complex data.

For this captcha solver application that we implemented, we decided to use the regularized logistic regression algorithm. The regression algorithm still produces high probability of accuracy and is effective in data that are not too complex. If our case, the unique captcha generated images are more closely related to each other than using data of handwritten letters and numbers.

In current research and professional projects, such as Andrew Ng's handwritten word recognizer application, most implement neural networks for their popularity, complexity, and potential. This approach could be used for future continuation of this project, but in our project this possibility can be discussed later.

2 Approach

First, to implement the main algorithm of the application, we need to load the data images. These data images need to be all of the same size in pixel dimension and be of characteristically unique. The Claptcha generator in Python would generate these multitudes of images that can be used for training data. The images generated are all unique, morphed in some way, and have minor discrepancies, such as lines going through the text. The deformities are characteristic of captcha images. As a note, Claptcha can also generate the images later to be used for testing and predicting. This is an example captcha image of the letter A,



The Claptcha images generated are all originally distorted. We implemented morphological operations to clean up the image and remove unneeded characteristics, such as the random slashes through the letter or number. Some of the morphological operations that can be used are closing and opening. Once these images are processed and ready to be used for training, we save them to a data structure, such as a struct, with a vector of the pixel values and a class label representing the image character.

For training these images, we attempt the regularized logistic regression. In the example of training digits, there would be a specific label for each digit. The class label for digit 1, would be class label 1, and so forth. The first approach is to see if we can compute the cost function for an unregularized logistic regression. The reason for this is that the majority of the unregularized cost function is also used in the regularized logistic regression. The formula for the cost function in unregularized logistic regression is

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

In this formula, the *m* is the number of images being trained, *y* is the matrix of size *m* by 1, whose values are set to the class label of the image, *X* is the matrix of size *m* by the number of total pixels in each image, and h_θ is the hypothesis of the specified row in *X*. In simpler terms,

$$h_\theta\left(x^{(i)}\right) = g\left(\theta^T x^{(i)}\right) \text{ and } g(z) = \frac{1}{1+e^{-z}}$$

The cost function must be minimized to find the best result that matches with the data given. The gradient descent algorithm is used to minimize the cost function by changing the θ to reduce the cost value **J**. The formula used to calculate the gradient function is

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

The cost function is an algorithm used to find the best line that fits and represents a group of data. In our case, this is not enough to be used as a representation of our data. The unregularized cost function is for the case when there is only one class label. For character recognition, there are many class labels representing each digit and letter. For this case, we took the next step of implementing the multiple regularized logistic function. This is part of the one-vs-all classification. Instead of treating the data with multiple class labels, we prioritize the data with one focused class label at a time and everything else is ignored as "other data". The cost function will calculate a line where the positive side of the line represents the data with the prioritized class label and the negative side of the line represents the rest of the unimportant data.

In this case, we begin to regularize the cost function, by regularizing the θ in a defined number of iterations. Thus the cost function will now be formatted to look like

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

In consequence, the gradient calculations will also be changed

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}\right) \qquad for\, j = 0$$

$$\frac{\partial J}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}\left(\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)}\right) + \frac{\lambda}{m}\theta_j \qquad for\, j \geq 1$$

In essence, the cost function is being called for each class label over a certain number of iterations. The final result of the cost function will be regularized to the point of convergence. The resulting θ will be used for the prediction testing for the one-vs-all classification. The input captcha image will compare to the theta probabilities calculated. The highest probability with the corresponding class label will be the prediction of the identity of the unknown letter or number.

3 Results

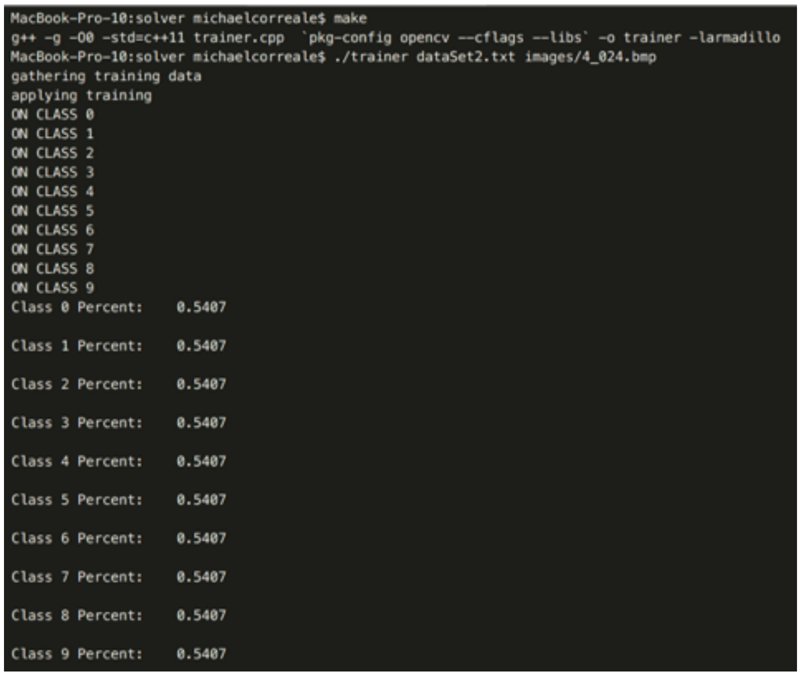
While many implementations of regularized logistic functions are done in matlab and other programming languages focused on mathematical algorithms, we decided to implement this in C++. The difficulties in completing this project in C++ is due to that there are not that many useful libraries and functions that can be used to calculate the algorithms given in the Approach section. There were many other difficulties and problems that we faced and we will discuss them further.

For creating the training images, we used the Claptcha generator as stated before. We decided to focus on digits first for ease of testing. Our training data includes 100 images of each class label, in other words, 100 images of each digit from 0 to 9. Each image is of 100 by 100 pixels. The image is larger than we desired to be, because when generating smaller images with the Claptcha generator, it would output images where the characters and digits were clipped. 100 by 100 pixel images had some minor clippings, but we decided that it was minor enough for the letter and or digit recognition to still work.

To clean the captcha images form the unneeded deformities (lines crossing through the characters), we implemented a binary threshold and morphological operations. For threshold, we used the defined value of 175, as it gave the best result of the image. Then to remove the lines through the characters, we used the closing morphological operation with a 5 by 5 kernel, where the immediate corners are set to 0.

Our first implementation of unregularized logistic regression seemed to be correct. It produced a cost function and gradient values that seemed reasonable. We also depended on the library Armadillo to provide us with matrix and linear algebra functions. When we applied the regularization additions to the cost function and gradient descent calculations, we faced many issues and confusion.

When we ran our application, the main problem we faced was that the gradient function was not calculating correctly. For each label we processed, the theta probability was increased. And if we tried to edit or change the matrix multiplication, we would have wrong results for the different class label



```
MacBook-Pro-10:solver michaelcorreale$ make
g++ -g -O0 -std=c++11 trainer.cpp  `pkg-config opencv --cflags --libs` -o trainer -larmadillo
MacBook-Pro-10:solver michaelcorreale$ ./trainer dataSet2.txt images/4_024.bmp
gathering training data
applying training
ON CLASS 0
ON CLASS 1
ON CLASS 2
ON CLASS 3
ON CLASS 4
ON CLASS 5
ON CLASS 6
ON CLASS 7
ON CLASS 8
ON CLASS 9
Class 0 Percent:    0.5407

Class 1 Percent:    0.5407

Class 2 Percent:    0.5407

Class 3 Percent:    0.5407

Class 4 Percent:    0.5407

Class 5 Percent:    0.5407

Class 6 Percent:    0.5407

Class 7 Percent:    0.5407

Class 8 Percent:    0.5407

Class 9 Percent:    0.5407
```

probabilities. This is an example output,

Clearly, the way we calculated the gradient function is incorrect and the saving of the theta calculated from minimizing the cost function is incorrect.

4 Discussion

From the difficulties of producing this application in C++, our first realization is that it would have been much easier to implement this in matlab. Otherwise, our main issues were from the incorrect application of the regularized logistic regression calculations. More specifically, we had troubles minimizing the cost function, as well as the gradient decent, which lead to ill-conditioning problems in our matrices. While we understood the mathematics behind the algorithms, the implementation was very difficult for us.

When looking at Andrew Ng's application of the regularized logistic application, we could see many of our mistakes and differences. The application of calculating the minimization of the cost function was very different as he used another function that calculated the convergence of θ with an abstract algorithm with no explanation. In our case, we decided to calculate the convergence by defining a set number of iterations. This difference is clear that we are unsure how to converge and for how long. The convergence of θ is important to output accurate class label predictions. Future work in modifying the gradient descent algorithm would be needed to improve the results and predictions of captcha images in our application.

5 Conclusion

Regularized logistic functions have been an important algorithm in the studies in machine learning and computer science. Our application used this algorithm for the real life problem of solving captcha images. Many steps were implemented to solve this problem, including morphological operations, cost functions and gradient descent algorithms. A full understanding of linear algebra, machine learning, and image recognition is needed to produce a valid solution. As a conclusion, the main complexities of regularized logistic regression are in the cost functions and gradient descent algorithms. The mastery in these algorithms can lead to the perfection of a captcha solver and other applications like picture and handwriting recognition. There are high hopes that it will become something more fruitful in the near future.

References

N. (2016, March 31). NASaCJ. Retrieved December 16, 2017, from http://nasacj.net/

Ng, A. (n.d.). *Machine Learning*. Lecture. Retrieved December 16, 2017, from https://www.coursera.org/learn/machine-learning/home/welcome.

Yin, L. (n.d.). *Visual Information Processing*. Lecture. Retrieved December 16, 2017, from http://www.cs.binghamton.edu/~lijun/CS555_Fall2017/2017Fall_CS555.html

Our github link:

https://github.com/mcorreale1/captchasolver