UNIVERSITY OF CALIFORNIA

Los Angeles

Classification with Hash Collision Networks

A dissertation submitted in partial satisfaction
of the requirements for the degree
Master in Applied Statistics

by

Christian Siyao Gao

2018

ABSTRACT OF THE DISSERTATION

Classification with Hash Collision Networks

by

Christian Siyao Gao
Master in Applied Statistics
University of California, Los Angeles, 2018
Professor Ying Nian Wu, Chair

The dissertation of Christian Siyao Gao is approved.

Qing Zhou

Mark S. Handcock

Ying Nian Wu, Committee Chair

University of California, Los Angeles

2018

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

(Acknowledgments omitted for brevity.)

VITA

2011–2015 M.S. (Statistics), UCLA.

CHAPTER 1

Introduction and Related Work 3 Pages

1.1 Problem Definition- Supervised Classification

-Kernel Trick -Approximate Neared Neighbors -Hamming Distance Locality-sensitive hashing

1.2 Question To Be Answered

-Many top image classification algorithms today use neural networks for training filters and use hashing tricks in order to estimate hamming distances when the dataset has alot of samples and classes.

-Can we skip neural networks entirely and rather use a network of hashmaps to create the same filters and generate the same results? Computationaly simple memory based methods

CHAPTER 2

Related Work 2 Pages

2.1 Image set representation and matching

2.2 Deep learning for single image hashing

Hash Map tricks for Spam Filters and Image Retreival. Previous work using hashing functions with CNN.

2.3 Deep learning for set classification

2.4 Hashmap development and large scale data retreival

Hbase White paper

CHAPTER 3

Method Overview

3.1 Overview of Notations

Our approach classification without using gradient based methods is to create multiple hash representations of the training features and then to store these features mapped to its original class in the training set. When predicting an objects class we create the same hash representations of the objects features and count the number of collisions. By storing the outputs each hashfunction in its own hashmap or node, we can observe the performance of each individual hash function to train filters within the hash function rather than using a CNN or convoluted neural network. A hash node N has 3 parts- one hash function, one hashmap, and a prediction function.

A typical hash function $h(x)$ or $h(x; S_n, F_n, T_n)$ will have 4 components where x is the training observation, S_n is a sampling parameter describing how to break down an observation into smaller pieces, F_n is a filtering parameter describing how to filter emphasize certain features of the data, and B_n is a Bootstrap parameter a non-random way to parametrically generate more versions of the data to get more hash keys if the training set is limited. Given an observation X_i a hash function will generate hashes: $\{h_{1:n}, h_{2:n}, ...h_{i:n}\}$ where i corresponds to the index of the observation and n respresents the corresponding hashnode. In the simple case, the hashes are then stored in a hashmap $\{h_{i:n}$ (generated from $h(x)$): $C_i\}$ where $h_{i:n}$ is the row key and C_i is the class of the observation from the training set stored as the value to the key. A prediction function $pred(x)$ is then used to generate prediction output from a new observation x from the test set. Thus we can model the probability of each observation to be of a certain class as:

$$P(C = c|x_i) = \frac{\#\text{ of Values } (V) = c}{\#\text{ of Values } (V) \neq c} \tag{3.1}$$

Where V is class values attained when quering a hashmap $M(h)$ from the prediction hashes generated: $\{h_1, h_2, h_3...\}$

3.2 Overview of Process

The Overall Method is:

1. For dataset $X = \{x_1, x_2, ...x_i\}$ generate hashes $H = \{h_{1:n}, h_{2:n}, ...h_{i:n}\}$ from $h(x)$ in node $N : \{N_1, N_2, ...N_n\}$ with arbirary or randomly generated paramters S_n, F_n, T_n

2. If data is small, generate $H' = \{h'_{1:n}, h'_{2:n}, h'_{3:n}, ...\}$ through parametric bootstrap of the original hash set H for each node and store the in the corresponding node's hashmap.

3. Predict $\{c_1, c_2, ...c_i\}$ classes from each hash node

4. Do a simple vote between each of the hashnodes for the final prediction = $C_{i;pred}$ or use a boosting method to aggregate the predictions

5. Rank each node in terms of performance and calculate the correlations between the predicts of each node to create a set of N top nodes combining or removing nodes that are too similar or perform badly.

6. Add and train new nodes with randomly set filters and parameters as neccessary

CHAPTER 4

Method Overview 6 Pages

Outline of hash-collision neural networks.

CHAPTER 5

Training 3 Pages

Training hash-collision Network

CHAPTER 6

Predicting 3 Pages

Prediction hash-collision Network

CHAPTER 7

Training and Predicting Mnist

Filtering and Methods Stuff

CHAPTER 8

Experimental Results

Results of Experiment

CHAPTER 9

Scalability and Performance

Performance

CHAPTER 10

Conclusion

List of Results and Graphs