Protecting neural networks against black box attacks

Student: Rhys Patten, Supervisor: Professor Peter Flach

University of Bristol, Department of Computer Science

Introduction

Black box attacks on neural networks can be used to store arbitrary information in neural networks by adding synthetic images to training data. The labels these images have encode a bit string, and because the synthetic data can be reproduced so can the bit string that is the concatenation of all of their labels. This project aims to provide practical methods of analysis for existing systems and systems being developed to minimize information leakage via machine learning.

This project covers:

1. Analysis of memorization in neural networks to find an effective way of detecting featureless data among data with features.
2. Weight analysis within networks as a method of detecting possible synthetic data.
3. Genetic algorithms for developing network architectures that are robust to arbitrary memorization.
   - Analysis of different network shapes on memorization capacity
   - Experimenting with domain specific architectures to find an optimal layout for learning the MNIST data set on a two hidden layer network.

1. Project Outline

The goals of this project are as follows:

- To understand and document the current state of research and how this can be used to assist in solving the issue outlined.
- To find some approaches/ideas that appear suited to this problem that are yet to be tried, and adapt methods that have been proven to make them more effective.
- Running suitable and effective experiments designed to test these various factors.

2. Previous work

- Attacks on neural networks.
  - Black box attacks
  - White box attacks
- Memorization in neural networks.
  - Analysis of regularization techniques.
  - Analysis of capacity.
  - Experiments on different data sets.
- Differentially private models for neural networks.
- Genetic algorithms for developing neural net architectures.
  - Genetic algorithms for optimization of network architecture.

3. Preliminary Results

Some of my results so far:

Here I trained 10 networks each with 2 hidden layers on MNIST, with each connection removed with probability 0.5. All the networks achieved 97% accuracy on MNIST data but their accuracy on random images with random labels is as follows. Since I found an alarming variety in effectiveness of different random architectures I retrained each network with 5 different initialisations to see if this made any difference. I used random images/labels pairs from a consistent seed to reproduce the same synthetic data each time.

Below the first table I have shown the performance of 2 fully connected networks, one the same shape as the original and one with the same number of weight variables.

		Random data accuracy					
Net number	Connections in Network	1	2	3	4	5	Average
1	334,773	53%	69%	70%	49%	62%	61%
2	335,013	44%	55%	42%	14%	42%	39%
3	334,697	95%	43%	43%	60%	67%	62%
4	334,200	51%	53%	42%	85%	11%	48%
5	334,621	48%	61%	69%	89%	70%	67%
6	335,078	18%	44%	51%	54%	52%	44%
7	334,077	20%	55%	35%	41%	46%	39%
8	334,471	45%	19%	62%	56%	64%	49%
9	334,556	47%	67%	56%	62%	51%	57%
10	334,619	69%	84%	11%	67%	82%	63%

		Random data accuracy					
Fully connected size	Connections in Network	1	2	3	4	5	Average
784x512x512x10	668,672	11%	11%	11%	11%	10%	11%
784x304x304x10	334,199	30%	15%	11%	11%	11%	16%

Results show that weight initialisation plays a key role in possibly in whether a network can memorize data at all but certainly in how long it takes it to memorize that data. These results also suggest that the layout of connections within the network could also effect the rate at which or whether random data can be memorized. These act as random masks over weight variables, so these are likely linked. Current research suggests that in repeating for large numbers of weight initialisations could be a solution to this, but to do this I would need to find a much more effective method of experimentation.

4. Progress and Status

Current Issues:

- Weight initialization seems to be an issue, there seems to be noticeable variance in performance of different architectures

Aims and open questions:

- Are different random architectures actually better/worse or is this random noise only from initialisation?
- Can we determine a way of initialising weight vectors such that they memorize better/worse?
- Currently only random labels/images are being looked at - could different distributions of synthetic data effect its ability to be memorized?
- Can we use grouping of weights of featured and non featured images as a way of detecting groups of synthetic data?
- Previous research suggests that taking 100 initialisations per network was their way of dealing with this issue, this could allow for cleaner statistics but would make performing current experiments infeasible. Is there a sensible way around this?

Project completion:

- To be able to effectively detect and possibly remove synthetic data from training data automatically
- To develop techniques to training networks that are resilient to attack