**Adversarial Machine Learning – Home Assignment 6 Part I**

Goal and Objective

Implement a white box version of the Hop-Skip-Jump-Attack

The first part of the home assignment will be dedicated to the basic elements of the attacks, while in the second part you will be required to assemble the different parts together.

Assignment Steps

1. Train a simple MNIST convolutional classifier

2. Randomly choose two images out of the MNIST dataset so that each image belongs to a different class

3. Traversing Hyper Lines in Input Space

a. Interpolate the two images and print out 11 images that exist on the hyper-line connecting the original two images, as follows –

<original image 1>,<interpolation 1>, <interpolation 2>, …,<interpolation 9>, <original image 2>

For each of the 11 images print out the class label assigned to it by the original classifier. The result is demonstrated below for images 18 and 19 in the MNIST test set. See how the first image morphed into the second (the number on the title of each image is the relevant class label) –



b. Explain your results –

i. What does it mean to “interpolate” the two original images?

ii. How is that related to the hop-skip algorithm we have reviewed in class?

iii. Where is the decision boundary located (roughly) with respect to two original images and the hyper line that connects them?

4. Implementing the Binary Search Algorithm

a. Implement a Binary search algorithm that finds the classification boundary between two input images

**Hint**: You should stop the search when the length of the search domain becomes lower than some small scalar value (e.g. 1E-7)

b. **Bonus question:** Implement a Binary search algorithm that can operate concurrently over multiple image pairs without an explicit loop for iterating through the different pairs

Such an implementation in order to allow efficient TF based implementation later on **Hint:** Calculate an indicator function using the built in numpy/TF ‘==’ operator. Compare the class associated with the middle point to that of the source point, then use this indicator value in order to compute the min/max points for the next iteration

5. Gradient Calculation

As we discussed in class, Hop-Skip-Jump aims to find the normal to the boundary surface (the perpendicular vector) at each iteration. However, being a black box attack this is done by approximating the gradient at the boundary point.

We will be implementing a white box version and can hence calculate the gradient directly.

Write a short function to return the relevant gradient.

Much like C&W the Hop-Skip attack uses the following formula –

�� = ��������������(��) − max

��≠����������������(��)

Don’t forget ��(��) is a vector. Extracting a specific element ����(��) requires some thought.

Your function should return ����

����

Submission guidelines

As always, submit full code as well as a single word document with the results and description. Enjoy.