Ejemplos de adversarios en Deep Neural Redes.

Edmundo Manuel Miranda Bailón
National University of Engineering
Lima, Perú
emirandab@uni.pe

Elias Josue Llampi Aliaga
National University of Engineering
Lima, Perú
elias.llampi.a@uni.pe

José Manuel Yoichi Rios Yamamoto
National University of Engineering
Lima, Perú
jriosy@uni.pe

I. Introducción

En la actualidad, en el desarrollo continuo de métodos de desarrollo en Machine Learning, uno de sus modelos, las Redes neuronales, las cuales su desarrollo ha dado buenos frutos como, por ejemplo, su aplicación en el procesamiento de datos, el reconocimiento de voz, la robótica o su uso para los engañosos Deepfake; sin embargo, este modelo posee una vulnerabilidad y esa debilidad son los ejemplos adversarios.

Un ejemplo adversario es una entrada a la red neuronal que deriva en una salida incorrecta, en otras palabras, clasifica mal la entrada, y los ejemplos adversarios su propósito es el engañar o confundir a una red neuronal.

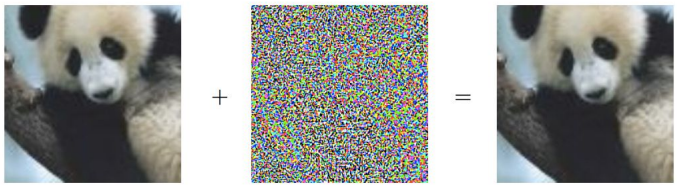

Figura 1. Ejemplo adversario para engañar una red neuronal y tome un gibón por un panda

Aquí podemos observar que mediante un ligero agregado de *ruido* a una imagen que la red neuronal ya reconocía, ahora obtenemos una respuesta diferente.

En los objetivos de los ejemplos adversarios están las redes neuronales aplicadas a [3]:

- Visión por computadora:clasificación de imágenes, detección de objetos.
- Procesamiento de lenguaje natural: traducción automática, generación de texto.
- Seguridad del ciberespacio: servicios en la nube, detección de malware, intrusiones en la red.
- Físico: reconocimiento de señales de trafico, cámaras de suplantación, reconocimiento de rostros.

En los ejemplos adversarios nos encontramos con 2 tipos de ejemplos, aquellos que simplemente buscan cualquier entrada para que la red neuronal se pueda confundir y se obtenga la salida con la cual se diseña el ejemplo adversario, estos son llamados ejemplos **no dirigidos**, pronto veremos que estos son en su mayoría entradas sin sentido; y el otro tipo de ejemplo, los **dirigidos**, los cuales son construidos para parecerse a una entrada determinada pero logran que la red neuronal se confunda y de como resultado una salida diferente [4].

En este trabajo analizaremos una red neuronal programada en el lenguaje Python entrenada bajo una base de datos MNIST de forma que clasifique números de 28x28 pixeles, luego diseñaremos los ejemplos adversarios para identificar las vulnerabilidades de la red neuronal [1] y estudiaremos posibles defensas.

II. Red Neuronal Vanilla y Ejemplos Adversarios

Una red neuronal funciona como la representación matemática de las neuronas en el cerebro, capaces de modelar su funcionamiento en la ejecución de tareas complejas donde cada se tiene una capa de entrada y una de salida de forma básica donde la matriz de entrada se multiplica por un vector de pesos para obtener una predicción según sea el calculo.

Una red neuronal vanilla se compone de una capa de entrada, una capa escondida única y la capa de salida, esta capa escondida intermedia se encarga de hacer una multiplicación de pesos adicional para categorizar mejor la información ingresada. En verdad una red neuronal puede tener muchas mas capas intermedias escondidas.

En nuestro proyecto tendremos una red neuronal de tipo MNIST la cual se encargara de categorizar imagenes de numeros en una cuadricula de 28x28 donde debido a que debemos evaluar cada pixel entonces se tendrá una capa de entrada de 784 neuronas, una capa intermedia de 30 neuronas y una salida de 10 neuronas con la cual tendremos la categoría si el numero corresponde a un numero desde 0 a 9.

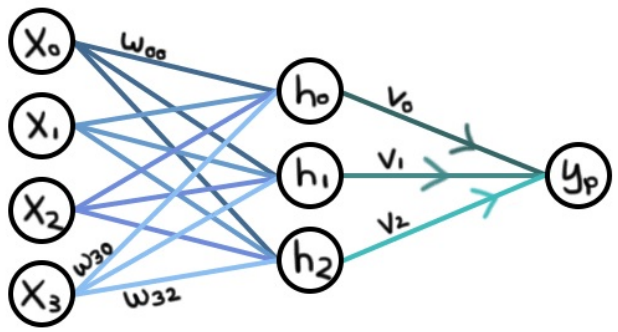

Figura 2. Estructura de una red neuronal vanilla

En el trabajo nos concentraremos en 2 tipos de ejemplos adversarios, aquellos que son no dirigidos con los que solo se busca que la red neuronal mande una falsa categorización sin importar cual haya sido la entrada, si es legible o no; y los dirigidos, aquellos que son generados para ser legibles para los humanos con una forma especifica pero que la red neuronal aun así mande una falsa categorización.

III. Estado del Arte

III-A. Los ejemplos adversarios no son errores

En el articulo **Adversarial Examples Are Not Bugs, They Are Features** [2] se menciona que en trabajos previos en el campo de muestras adversarias, se tiende a ver estas muestras como aberraciones que surgen ya sea de la naturaleza de alta dimensión del espacio de entrada o fluctuaciones estadísticas en los datos de entrenamiento, por lo que los objetivos se centraban en mejorar la robustez de los modelos aumentando la precisión del mismo.
Sin embargo, el artículo demuestra que los ejemplos adversarios se pueden atribuir directamente a la presencia de características no robustas: características (derivadas de patrones en la distribución de datos) que son altamente predictivas, pero frágiles y (por lo tanto) incomprensibles para los humanos. Después de capturar estas características dentro de un marco teórico, se establece la existencia generalizada en conjuntos de datos estándar. Finalmente, se presenta un escenario simple donde se vinculan rigurosamente los fenómenos que se observaron en la práctica a una desalineación entre la noción (especificada por el ser humano) de robustez y la geometría inherente de los datos.

III-B. Explaining and Harnessing Adversial Examples

Este paper nos relata que los primeros intentos de explicar el fenómeno de los ejemplos contradictorios se centraron en la no linealidad y el sobreajuste, pero se evidencia que la causa principal de la vulnerabilidad de las redes neuronales a las perturbaciones adversas es su naturaleza lineal. Esta explicación está respaldada por nuevos resultados cuantitativos mientras se da la primera explicación del hecho más intrigante sobre ellos: su generalización a través de arquitecturas y conjuntos de formación. Es más, esta vista proporciona un método simple y rápido de generar ejemplos contradictorios. Usando este enfoque para proporcionar ejemplos para el entrenamiento adversario, reducimos el error del conjunto de prueba de una red maxout en el conjunto de datos MNIST(Modified National Institute of Standards and Technology database). [1]

III-C. Ataques adversarios de caja blanca

En estos ataques, donde el atacante posee conocimiento total sobre el modelo de destino, incluidos los algoritmos de entrenamiento, la distribución de datos y los parámetros del modelo. Los atacantes identifican el espacio de características más vulnerable del modelo y luego alteran las entradas utilizando métodos de generación de muestras adversas.
En el artículo **Review of Artificial Intelligence Adversarial and Defense Technologies** [3] se muestra un ataque de este tipo, donde se emplearon redes neuronales convolucionales para el reconocimiento de señales de carreteras, y mediante métodos generadores de muestras adversarias se crearon muestras adversarias, que variaban de las originales en el ángulo de vista, distancia o imágenes superpuestas como graffitis o pegatinas sobre las señales. Dichas muestras adversarias dieron como resultado una clasificación errónea de los modelos cercana al 100 %.

IV. Metodologia

En esta sección se describen las herramientas y la metodología que se usarán en el presente trabajo:

IV-A. Archivos de librerías

Se hará uso de las siguientes librerías y archivos:

- **network.py**
  Con este archivo obtendremos el modelo para implementar el algoritmo de aprendizaje mediante la gradiente descendiente para una red neuronal prealimentada.
- **mnist.py**
  Con este archivo podremos cargar el archivo de imagen mnist 'mnist.pkl.gz' para poder separar los datos en 'datos de entrenamiento', 'datos de validación' y los 'datos de prueba'
- **numpy**
  Librería utilizada para el mejor manejo de matrices, funciones matemáticas y posteriormente modelar una función de optimización para poder generar los adversarios ya sean dirigidos o no dirigidos.

- **matplotlib**
  Librería utilizada para poder graficar de mejor manera nuestros resultados y evaluar los diferentes comportamientos de nuestra red neuronal entorno a las pruebas

Nuestro metodologia consistirá en:

1. El entrenamiento de la red neuronal para poder categorizar las diferentes entradas de números dibujados en una cuadricula de 28 x 28 cuadriculas.
2. El uso de la función de optimización:

$$C = \frac{1}{2} \left\| y_{goal} - \hat{y}(\vec{x}) \right\|_2^2 \tag{1}$$

   Donde y_{goal} es la salida que queremos conseguir y $\hat{y}(\vec{x})$ es la salida que obtenemos dado nuestro vector de entrada x compuesto de 784 componentes. Al minimizar esta función obtendremos la entrada a la red neuronal de los ejemplos adversarios no dirigidos.
3. el uso de la función de optimización:

$$C = \frac{1}{2} \left\| y_{goal} - \hat{y}(\vec{x}) \right\|_2^2 + \lambda \left\| \vec{x} - x_{target} \right\|_2^2 \tag{2}$$

   Al minimizar esta función obtendremos nuestra la entrada a la red neuronal de los ejemplos adversarios dirigidos. Donde x_{target} es la imagen que queremos a la cual queremos que se parezca nuestro ejemplo adversario y λ es un hiperparametro para poder especificar cual termino es mas importante, su valor se basa en prueba y error.
4. Estudiando los resultados obtenidos mediante la experimentación de la red neuronal probaremos un método para el cual la red neuronal pueda ser mas resistente a los ejemplos adversarios.

Referencias

[1] Ian J. Goodfellow, J. S. . C. S. *Explaining and Harnessing Adversarial Examples*. ICLR, 2015.

[2] Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial examples are not bugs, they are features, 2019.

[3] Shilin Qiu, Qihe Liu, S. Z. . C. W. *Review of Artificial Intelligence Adversarial Attack and Defense Technologies*. MDPI, 2019.

[4] Veerapaneni, D. G. . R. *Tricking Neural Networks: Create your own Adversarial Examples*. Machine learning at Berkeley, 2018.

Índice de figuras