Understanding and Generating Adversarial Examples

Kavinya Rajendran and Prasanth Priya Nesan Soundra Pandian, *Student, ASU*

Abstract— Injecting very miniscule distortion to the data, causing a classifier to misclassify a label is known as generating adversarial example. This has been a major concern of machine learning models, as it affects the accuracy of the classifier. This project focuses on building a 2-class classifier using logistic regression, generating adversarial examples and test accuracy degradation of the classifier for the adversarial examples. It is surprising that the classifier misclassifies these adversarial examples although the difference between actual and adversarial data is almost imperceptible to the naked eye. The reason was such misclassifications was initially believed to be the non-linear and overfitting nature of the data. In [1], it has been argued that it is due to their linear nature, with generalizing across architectures.

Index Terms—Adversarial Examples, Cost Function, Logistic Regression, Logistic Sigmoid.

1 Introduction

THE project aims to explore the idea of generating adversarial examples and understand how it reflects on the performance of the classifier. A model is built on logistic regression and its performance is studied with and without adversarial examples. It also extends to find the performance degradation for different levels of noise injected. Logistic regression is used as a classifier which classifies the images into either class 1 or class 7. The goal is to study the performance fluctuation for the different images. The images from MNIST digit recognition data [2] will be used as dataset for this project.

2 Methods

2.1 Model – Logistic Regression

The purpose of this method is to train a model to predict class labels $y \in \{-1,1\}$, with probability

$$P(y = 1) = \sigma(w^T x + b) \tag{1}$$

The $\sigma(z)$ is a logistic sigmoid function as given in [1]. During training, the cost function (J) is defined as a soft function given by,

$$J = \zeta\big(-y(w^T x + b)\big) \tag{2}$$

The $\zeta(z)$ is defined as $\log\big(1 + \exp(z)\big)$. The parameters of the model are the weights, w and bias vector, b. The model determines the best or optimal theta value (weight and bias), using the gradient function which is differential of equation (2) with respect to theta. Assume a part of differential to be z with, $z = -y(w^T x + b)$, the grad is given by,

$$\Delta J/\Delta\theta = \ 1/(1 + \exp(z)) \times \exp(z) \times -yx \tag{3}$$

2.2 Adversarial Example Generation

For the given problem statement, x is the input vector and y is its corresponding class label. We generate adversarial examples by injecting a minute distortion η to the actual input vector. This is given by,

$$\tilde{x} = x + \eta \tag{4}$$

The parameters of the function are the weights and bias vector, which together is considered as θ. Differencing equation (2) with respect to x would give the corresponding sign to be used for this injection of adversial image. The purpose of this method is to generate adversarial examples using the "Fast gradient sign method", with

$$\eta = \ \varepsilon \text{sign}(\nabla_x J(\theta, x, y)) \tag{5}$$

The $\nabla_x J(\theta, x, y)$ is given by differentiating equation (2) with respect to input vector x.

$$\Delta J/\Delta x = \ 1/(1 + \exp(z)) \times \exp(z) \times -yw^T \tag{6}$$

3 Protocols

Logistic regression is performed on the image dataset and model is built using the training data. The performance of the classifier is measured using the testing data, in which performance corresponds to amount of misclassification. On generating the differential of cost function with respect to x as given in (6), adversarial images are generated by introducting noise, where the noise η, is as represented in (5). The model built is again applied to the new set of adversarial images and then the performance of the linear classifier measured. The reduction in the performance of the linear classifier is observed. Project is carried out with 4 phases as mentioned below.

1. Understanding of the problem statement, gather data and implementation setup.
2. Learn model using logistic regression on training dataset from MNIST digit recognition data [2].
3. Generate adversarial examples by implementing fast gradient sign method.
4. Evaluate performance of testing data with and

- *Kavinya Rajendran is with the Arizona State University, Tempe, AZ 85281. E-mail: krajend2@asu.edu.*
- *Prasanth Priya Nesan Soundra Pandian is with the Arizona State University, Tempe, AZ 85281. E-mail: psoundra@asu.edu.*

without adversarial examples. Observe reduction in performance of the model.

Phases 3 and 4 are repeated with different values of ε.

4 Understanding Of The Project

As described in the earlier sections, the goal of this project is to generate adversarial examples that causes misclassifications of images even with smallest perturbation and to understand why classifiers do so.

According to [1], the adversarial examples are because the models are too linear and aligned with the weight vector, w. Structures with hidden layer can resist adversarial examples problem while linear models can't. The activation for this grows linearly by $w^T\eta$. It is noteworthy that linear models are forced to classify the adversarial example to the class that aligns most closely with its weights, due to which linear classifier are easily affected by them. As in (5), it is a good choice to generate adversarial examples because.

5 Tasks

5.1 Task 1 – Reason for linear classifiers getting easily affected by adversarial examples.

Linear classifiers get easily affected by adversarial examples due to the fact that they ignore all data below $1/255$ of the dynamic range. The adversarial examples shift the domain to a higher dimension making the linear classifier classify the points to a different class. The decision boundary remains the same, but the data points are shifted to a higher dimension. This makes the data points being misclassified as they make now lie closer to a different class than they normally do.

The weights and the direction of the decision boundary tends to misclassify these adversarial examples for a linear classifier as stated above.

5.2 Task 2 – Justification for having adversarial noise defined as $\eta = \varepsilon \mathbf{sign}(\nabla_x J(\theta, x, y))$.

Equation (4) makes use of differential of cost function with respect to the input vector x, making it a good choice for generating adversarial examples. The cost function itself is a good choice for the algorithm as more cost function indicates that numbe of misclassifications are more. This is the ideal characteristic of a cost function, which applies to our cost function as well. Following which, differentiating this cost function with respect to x, provides the direction (positive or negative) of drift from the decision boundary due to ϵ. Gradient can be computed using backprobagation in this case. This formulation of generating adversarial examples is also useful to speed up adversarial training.

5.3 Task 3 – Building Linear Classifier using Logistic Regression.

The dimensions of input feature vector and weights is increased by one to add the bias. After which, the logistic regression learns the ideal weights vector for the given data, starting from zero. This means that, the weights (θ), is initially set to 0, after which with a series of iterations, they tend to take better values that lowers the number of misclassified samples.

It uses the cost and gradient functions mentioned in equations (2,3) for a number of iterations to get optimal value of θ. As the end of iterations, the model would learn the ideal value for the weights vector given by θ.

5.4 Generating Adversarial Examples

Adversarial examples are generated for ϵ values 0.0025, 0.025 and 0.5, based on equation (4). The generated examples are then used as testing dataset to test the accuracy of the model. Accuracy of the model denotes the number of misclassifications by the model. This phase is used to observe the difference in accuracy of the model due to insertion of a minute value in the image dataset.

5.5 Predicting and determining the accuracy.

The model predicts the class labels based on the equation (1). It makes use of the input vector and the learned θ value to predict the possible class label. This method is used for a variety of cases, like to check the accuracy of training dataset, testing dataset and adversarial dataset.

6 Problems Encountered

Incorrect differentiation of the cost function for gradient and noise generation leads to deceptive results. Initially, the logistic algorithm was implemented with a different cost function and the soft plus function as gradient. This made the model completely insensible as both cost and gradient were not related. On understanding the concept of cost function, it was clarified that the gradient is differential of cost function with respect to theta value. This made a huge change in the prediction of the model both for testing and adversarial examples.

Conversion of data from .mat extension to .csv using numpy.savetxt created an exponential feature set by default. Although this doesn't have impact on the learned model, this format is corrected during later session of project.

7 Evaluation and Results

On applying the learned θ value to compute the class labels of the adversarial examples generated with different values of ε, an experiment was carried out to determine percentage of correctly classified samples. The experiment results are recorded as given in Table 1.

TABLE 1
Model accuracy for different dataset

Dataset	Accuracy
Training Sample	80.45%
Testing Sample	81.46%
Adversarial Example, $\epsilon = 0.0025$	79.38%
Adversarial Example, $\epsilon = 0.025$	47.80%
Adversarial Example, $\epsilon = 0.5$	47.53%

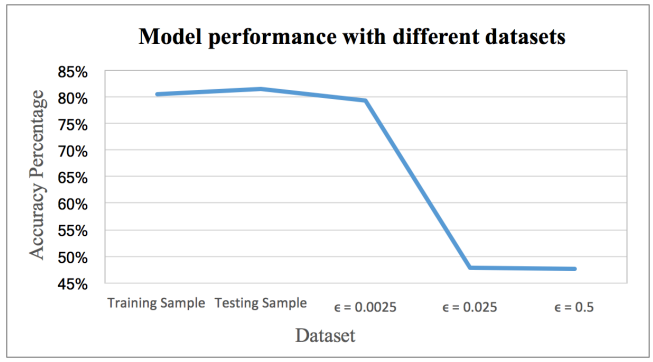

Fig. 1. Accuracy of the logistic regression model for each dataset.

Fig. 1 depicts a plot of the accuracy degradation of the model due to adversarial examples with different values of ε.

Fig. 2. Image for testing and adversarial examples with values 0.0025, 0.025 and 0.5 in sequence.

Fig. 2 demonstrates the generated adversarial examples for testing dataset and dataset with epsilon values 0.0025, 0.025 and 0.5 in sequence.

8 Conclusion

A linear classification model is built using logistic regression with a good accuracy. On testing the model with adversarial examples, it was observed that the accuracy of the classifier decreased. The greater the noise or ϵ, the more is the drop in accuracy. Thus, generation of adversarial examples and their effects on a linear classifier using logistic regression is understood clearly and observations are noted.

Acknowledgment

The authors wish to thank Professor Biaxin Li and Teaching Assistant Parag Chandakkar for their support and guidance throughout this project.

References

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," arXiv preprint arXiv:1412.6572, 2014.

[2] "MNIST digit recognition data," *https://drive.google.com/file/d/0B_k-8zTA1yWLYzEwUTlHaFVZa0k/view? usp=sharing*

[3] "Classifying MNIST digits using Logistic Regressio," http://deeplearning.net/tutorial/logreg.html