Fool me once, shame on - shame on you. Fool me - can't get fooled again.

Gabriel C-Parent
Département d'informatique et recherche opérationnelle
Université de Montréal
gabriel.c-parent@umontreal.ca

Dora Fugère
Département de mathématiques et de statistique
Université de Montréal
dora.fugere@umontreal.ca

Abstract

Adversarial examples generation from input space in neural network has shown that these powerful constructs can be manipulated into misclassifying previously well classified examples by adding an imperceptible amount of distortion. Using this methodology, we investigate the relative robustness of simple classifiers.

1 Introduction

Recently, neural networks have been brought under questionning. The smoothness assumption, the idea that imperceptible distortion of input shouldn't change the output was shown not to hold [1]. This is a remarkable finding since smoothness was assumed to be a necessary property of the learning process. This comes in stark constrast with feats such as automatic image description [2] and large-scale multi-character text recognition [3] to name but a few.

As for most real-world problems, there are many desirable and often conflicting goals when using machine learning. Amongst them speed, accuracy and simplicity are easy to justify. We'll focus on comprehensibility, because that justifies us using a simpler model. We interpret simplicity as "given two models with the same generalization error, the more comprehensible one should be preferred" [4]. This obviously is dependent on multiple other factors (e.g. speed and accuracy) but it does sound like the *keep it simple stupid* rule of thumb. Furthermore, as stated in [5], empirical comparison of performance is very context-dependent and can be infludenced by treatments such as the preprocessing steps, training parameters and model hyperparameters.

Inspired by the methodology to induce misclassification, we wondered if a similar optimization procedure could be applied to generate adversarial examples in a simpler classifier. For this purpose, we used a linear support vector machine (SVM). The only other contender would have been Naive Bayes, but we happen to like sklearn's implementation of the linear SVM [1] [6]. We report the robustness of our optimization procedure, the results on classifiers trained with different loss and regularization parameters and we then feed the generated adversarial examples to a neural network to see if some underlying feature of the image was captured.

[1] we wouldn't risk reinventing the square wheel

2 Framework

2.1 Dataset

The experiments were performed on the MNIST dataset [7].

Let $X = \{0, 255\}^{784}$, the input domain. This is the set of 28×28 8-bit image.

Let $Y = \{0, 9\}$, the output domain. This is the set of valid classes for an MNIST digit.

2.2 Preprocessing

The MNIST dataset was deskewed and brought back to 8-bit data. This allowed improvement performance for the linear SVM.

2.3 Optimization goal

Let $f : X \to Y$ a classifier mapping $x_i \in X$ to $y_i \in Y$.

We aim to solve the following optimization

$$\begin{aligned} &\text{minimize} && \|r\|^2 \\ &\text{subject to} && \\ & && x_i + r \in X \\ & && f(x_i) \neq f(x_i + r) \end{aligned} \tag{1}$$

This is quite similar to [1] but the newly generated images remain 8-bit to stay in the input domain of the MNIST dataset. Sadly, this also makes it a discrete optimization problem.

2.4 Optimization goal for the linear SVM

Suppose we want to misclassy an arbitrary image x_i correctly classified as y_1 by adding a vector r of distortion in a two-class setting.

The classifier classifies the input based on the following decision function.

$$y_i = argmax(x_i \cdot W^T + b) \tag{2}$$

The difference between the class weights of the classifier is W_{diff}.

$$W_{diff} = W_2 - W_1 \tag{3}$$

The distance between the values of the two classes is d.

$$d = x_i \cdot W_{diff}^T \tag{4}$$

To cause misclassification, r must respect the following constraint:

$$r \cdot W_{diff}^T > d \tag{5}$$

What we need is to find the smallest $\|r\|^2$ that will cause misclassification. Note that when there are more than two classes, we just apply the procedure to all other classes $y_i \neq y_1$ and choose the one with minimal squared euclidean norm.

2.5 Knapsack problem and the greedy approach

The problem is similar to the bounded multiple-class binary Knapsack problem [8], with the difference that we are searching for the smallest knapsack holding a value superior to d (equation 4).

The exact algorithm would be too costly for our purpose so we chose to use a greedy heuristic inspired by Dantzig's [9].

3 Experimental results

3.1 Precision of optimization procedure

Although the bounds on the optimization procedure could be arbitrary big, it is usually small for this problem, because d is usually quite big relative to the size of weight increments. This means tight bounds on the possible true value of the squared euclidean norm.

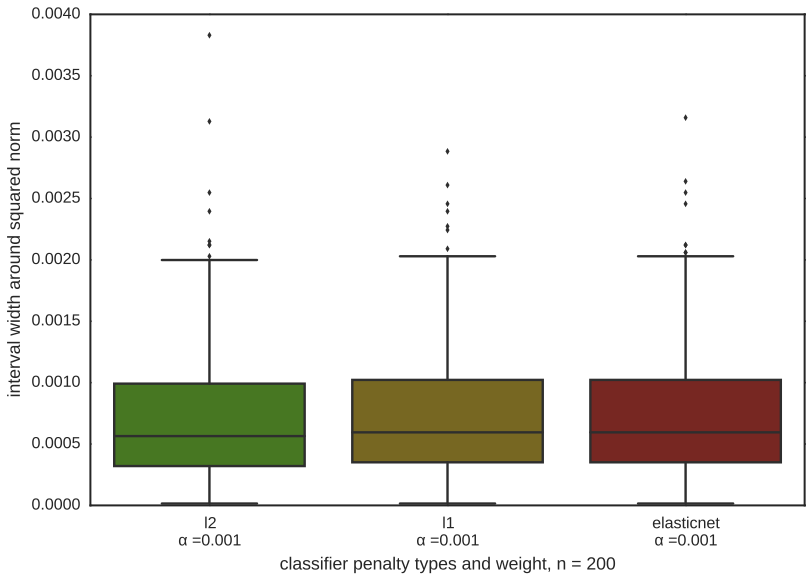

Figure 1: Size of the interval on the true value of the minimal squared norm for various classifier on the MNIST dataset. The bound is shown to be very tight.

3.2 Regularization schemes

We wanted to observe the effects of different regularization methods and parameters on the generation of adversarial examples. To do so, we started with a pilot run, to assess the potential of the various methods.

4 Discussion

As expected, having bigger weights allows the generation of adversarial examples with less squared euclidean norm.

Acknowledgments

We would like to thank caffein.

References

[1] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. URL `http://arxiv.org/abs/1312.6199`.

[2] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *arXiv preprint arXiv:1411.4555*, 2014. URL `http://arxiv.org/abs/1411.4555`.

[3] Ian J Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Sacha Vinay. Multi-digit number recognition from street view. 2013.

[4] Pedro Domingos. The role of occam's razor in knowledge discovery. *Data mining and knowledge discovery*, 3(4):409–425, 1999. URL http://link.springer.com/article/10.1023/A:1009868929893.

[5] David J. Hand. Classifier technology and the illusion of progress. *Statistical Science*, 21(1): 1–14, February 2006. ISSN 0883-4237. doi: 10.1214/088342306000000060. URL http://projecteuclid.org/Dienst/getRecord?id=euclid.ss/1149600839/.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[7] Yann Lecun and Corinna Cortes. The mnist database of handwritten digits. 1998.

[8] Francois Vanderbeck. Extending dantzig's bound to the bounded multiple-class binary knapsack problem. *Mathematical Programming*, 94(1):125–136, December 2002. ISSN 0025-5610, 1436-4646. doi: 10.1007/s10107-002-0300-7. URL http://link.springer.com/10.1007/s10107-002-0300-7.

[9] George B Dantzig. Discrete-variable extremum problems. *Operations Research*, 5(2), 1957. doi: 10.1287/opre.5.2.266.

4.1 example of the greedy heuristic

We encode the cost of adding a bit of distortion to a pixel in the form of a volume. For example, given vector of distortion r = [0, 3], the costs of adding 1 unit of distortion for each position of r is [1, 7]. For the values, we use the absolute value of vector W_{diff}. If W_{diff} = [-5, 10], the $\frac{value}{volume}$ ratios would be [5, $\frac{10}{7}$]. The best choice to improve the value

We iteratively choose the elements with the best until the critical value is d is passed.

At iteration n, the choices made are optimal of the knapsack of size $K_n = \sum_{i=1}^{n} w_i$. This is the case because every object in the knapsack was chosen with maximum $\frac{value}{volume}$ ratio and the knapsack is filled perfectly.

Let (K_1, v_1) and (K_2, v_2), the size and value of knapsacks at two iterations, we know that if $K_1 < K_2$ then $v_1 \leq v_2$, that is to say it's a monotonically increasing function. The inverse is also true, if $v_1 < v_2$ then $K_1 < K_2$.

For our purpose, this means that the true value of the norm $\|r\|^2 \in [K_{n-1}, K_n]$ if the algorithm finishes at iteration n. This boundary can be very small or very big, depending on the size of the weight increments. We measured the boundary size to show that in this problem, we have very little incertitude.

4.2 regularization methods and minimal squared euclidean norm