Title: Explaining and Harnessing Adversarial Examples

Authors: Ian J. Goodfellow, Jonathan Shlens and Christian Szegedy

Link: https://arxiv.org/abs/1412.6572

Long form summary:

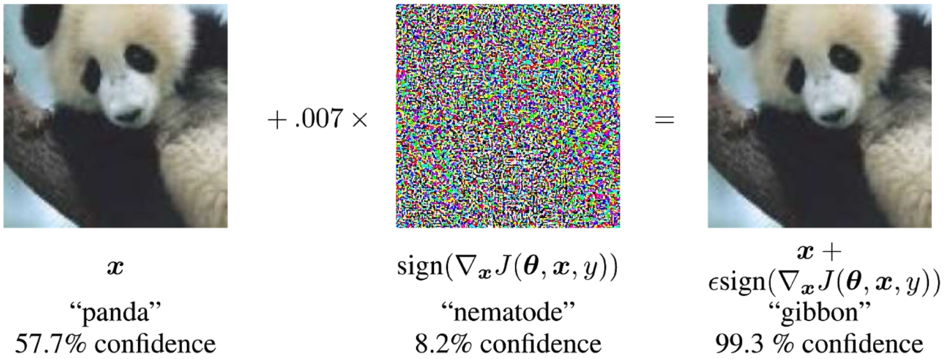
A bemusing weakness of many supervised machine learning (ML) models, including neural networks (NNs), are adversarial examples (AEs). AEs are inputs generated by adding a small perturbation to a correctly-classified input, causing the model to misclassify the resulting AE with high confidence. The existence of AEs highlights two general weaknesses of ML models, namely that they learn potentially misleading correlations, instead of the underlying features which define a class; and that they make over-confident predictions in regions where data is sparse.

Adversarial examples and linearity

Goodfellow et al. propose a linear explanation of AEs, in which the vulnerability of ML models to AEs is considered a by-product of their linear behaviour, combined with a high-dimensional feature space. In other words, small perturbations on an input can alter its classification because the change in activation (as result of the perturbation) scales with the size of the input vector. Thus the linearities which allow for efficient training are also a curse, because they make ML models susceptible to adversarial attacks.

Making an adversarial example

To advance their hypothesis, the authors introduce a method of efficiently generating AEs known as the “fast gradient sign method” (FGS). Given a cost function $J(\theta, x, y)$, where θ represents the current set of model parameters, x the input, and y the targets, we add a perturbation to x which is equal to the *sign* of the cost function gradient (with respect to x) times a constant ε. AEs produced in this way badly fool a maxout network. When $\varepsilon = 0.25$, 89.4% of AEs generated from the MNIST test set were misclassified, with an average confidence of 97.6%. The figure below shows an amusing example of a misclassified AE generated by FGS, this time by GoogLeNet on an ImageNet input.


$+ .007 \times$
$=$
$\boldsymbol{x}$
"panda"
57.7% confidence
$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence
$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

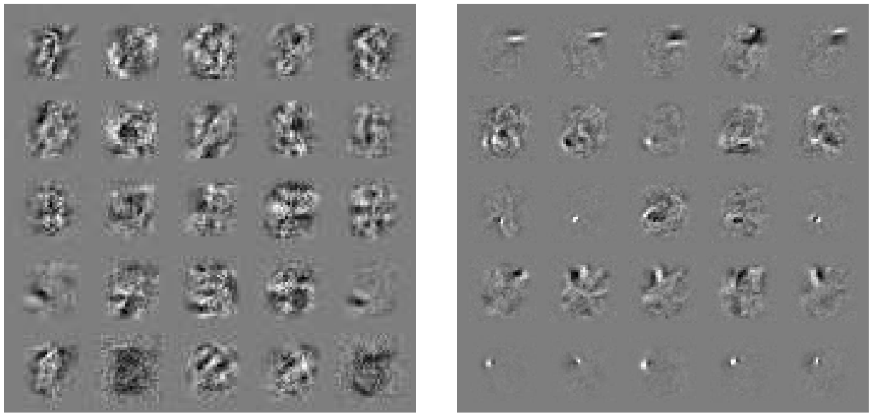
"Panda" + "nematode" = "gibbon"? Unlikely. The difference between the perturbed and unperturbed input is hardly even perceptible to the human eye.

Adversarial training to the rescue

As we saw above, identifying ways to effectively handle AEs is relevant to problems like image classification, where the input consists of intensity data for many thousands of pixels. To patch this blindspot we must train our models to recognize adversarials. One solution, explored by Szegedy et al. in 2014, is to mix AEs into the training set. While this does regularize the NN's response to AEs, it does not necessarily provide an advantage over dropout. Instead the paper suggests adding a term to the cost function $J(\theta, x, y)$ itself, which effectively penalizes the model if the cost for an FGS-generated AE is significantly different from that of the unperturbed input. Applying this form of adversarial training to a maxout network, the error rate on the MNIST adversarial test set drops dramatically from 89.4% to 17.9% ($\varepsilon = 0.25$). The network goes from misclassifying *almost every* AE to misclassifying less than 1 in 5.

One may ask whether adding random noise of a similar size as the FGS perturbation to training inputs would produce the same mitigating effect. The answer appears to be no. In the MNIST case, whether the authors randomly added $\pm\varepsilon$ to each pixel, or drew from a uniform distribution with range $(-\varepsilon, \varepsilon)$, the error rate and average confidence was similar to the results obtained without adversarial training.

Curiously, suppressing a network's sensitivity to adversarial perturbations may improve its interpretability. The authors demonstrated that maxout network weights trained on MNIST became more localized after adversarial training, and patterns corresponding to features of a handwritten digit were more discernible (See right panel of figure below).



Weight visualizations of maxout networks trained on MNIST. Each row shows the filters for a single maxout unit. Left) Naively trained model. Right) Model with adversarial training. For a guide to understanding these visuals, visit: https://ml4a.github.io/ml4a/looking_inside_neural_nets/

A unifying interpretation

Goodfellow et al.'s linear interpretation of AEs holds significant explanatory power. It provides a theoretical understanding of why simple models with nonlinear activation, such as shallow radial basis functions (RBF) networks are more robust to AEs than NNs. It also demystifies the observation that different linear models respond similarly to a given AE, even assigning the same incorrect class, despite being trained on different datasets. Through the linear view of adversarial training we also gain insight on general problems like interpretability and response to outliers, which suggests that understanding the existence of AEs is crucial for understanding ML models more broadly.