

Adversarial Training

Sargur N. Srihari

srihari@cedar.buffalo.edu

Level of understanding of DL

- DL networks can reach human level performance on an i.i.d. test set
 - But have they reached human level understanding?
- To probe this consider misclassified examples
 - Even networks at human level accuracy have a 100% error rate on examples intentionally constructed!

Adversarial example

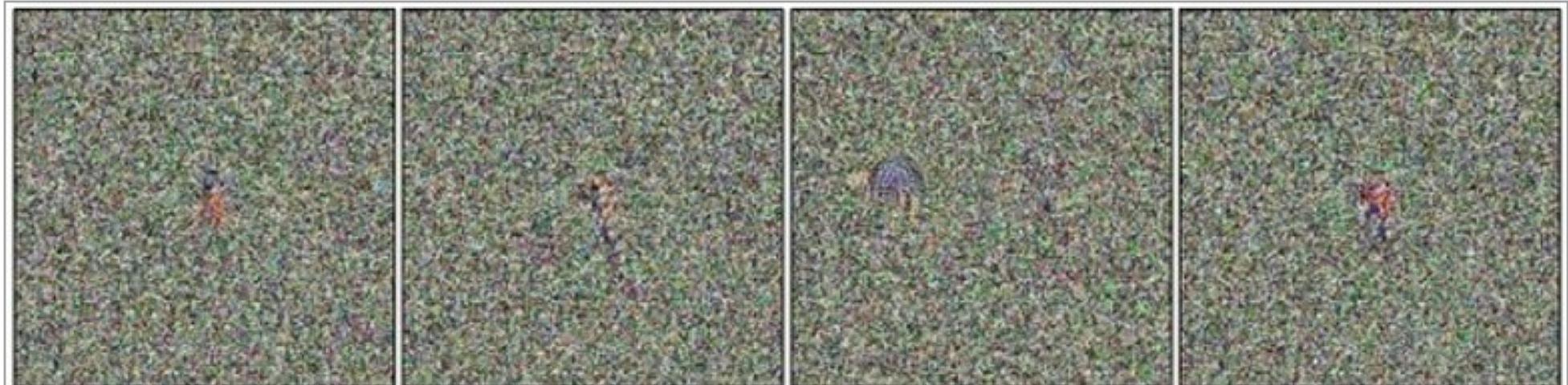
- ML models consistently misclassify adversarial examples
- Inputs formed by applying small but intentionally worst-case perturbations to examples from the dataset
 - Perturbed input results in incorrect answer with high confidence

Adversarial examples



Take a correctly classified image (left image in both columns), and add a tiny distortion (middle) to fool the ConvNet with the resulting image (right).

Some more examples



robin

cheetah

armadillo

lesser panda



king penguin

starfish

baseball

electric guitar

These images are classified with >99.6% confidence as the shown class by a Convolutional Network.

Cause of adversarial examples

- Primary cause is excessive linearity
 - Neural networks are built primarily out of linear building blocks
 - The overall function often proves to be linear
 - Linear functions are easy to optimize
 - But the value of a linear function can change rapidly with numerous inputs
 - If we change input by ϵ then a linear functions with weights w can change by $\epsilon\|w\|$ which can be very large in high-dimensional spaces

Generating adversarial examples

- An optimization procedure is used to search for an input x' near data point x such that the model output is very different at x'
 - In many cases, x' can be so similar to x that a human observer cannot tell the difference between the original example and the adversarial example
 - But the network makes a highly different prediction

Several methods for generating Adversarial examples

- Fast gradient sign method (FGSM)
- Basic Iterative method (BIM)
- Projected Gradient Descent (PGD)

FGSM Adversarial Example Generation

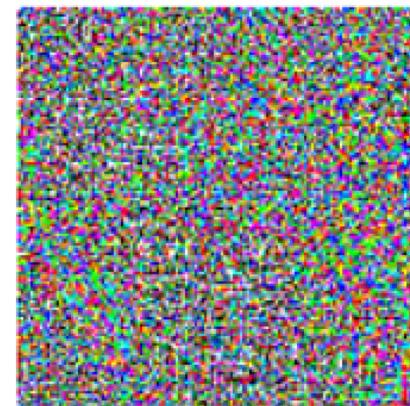
We add to x an imperceptibly small vector

Its elements are equal to the sign of the elements of the gradient of the cost function wrt the input. It changes Googlenet's classification of the image



\mathbf{x}

$$+ .007 \times$$



=



$$\text{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y))$$

$$\mathbf{x} + \epsilon \text{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y))$$

$y = \text{"panda"}$
with 58% confidence

$y = \text{"nematode"}$
With 8.2% confidence

$y = \text{"gibbon"}$
With 99% confidence

Adversarial Examples in Linear Models

- Precision of an input feature
 - With 8 bit resolution we discard info below $\varepsilon = \frac{1}{255}$
 - Need same response for x and $\tilde{x} = x + \eta$ for small elements of η , i.e., $\|\eta\|_\infty < \varepsilon$, i.e., max-norm constraint
- Consider activation $w^T \tilde{x} = w^T(x + \eta) = w^T x + w^T \eta$
 - Adversarial perturbation grows it by $w^T \eta$
 - Maximize increase by assigning $\eta = \text{sign}(w)$
 - For n features and $\|w\|_{ave} = m$, activation growth: $w^T \eta = \varepsilon mn$
 - Perturbation by η can grow linearly with n
 - For high dimensional problems, infinitesimal changes to input can add up to large change to output

$$\text{sgn}(x) := \begin{cases} -1 & \text{if } x < 0, \\ 0 & \text{if } x = 0, \\ 1 & \text{if } x > 0. \end{cases}$$

Linear perturbation of nonlinear models

- Linear view of adversarial examples suggests a fast way of generating them
 - LSTM, ReLU and maxout nets are designed to behave in linear ways, for easier optimization
 - More nonlinear models such as sigmoid networks are tuned to spend most of their time in the non-saturating, more linear regime for the same reason
- Thus cheap, analytical perturbations of a linear model should also damage neural networks

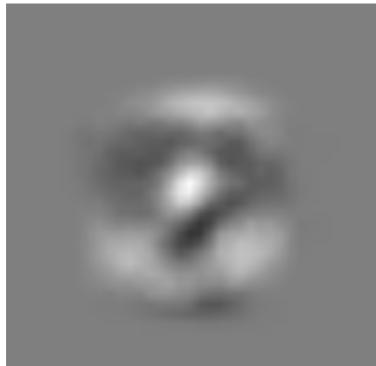
Fast gradient sign method of generating adversarial examples

- $J(\theta, x, y)$ is cost used to train network
 - θ are model parameters
 - x the input
 - y targets associated with x
- Linearize the cost function around the current value of θ , obtaining an optimal max-norm constrained perturbation of
$$\eta = \varepsilon \operatorname{sign}(\nabla_x J(\theta, x, y))$$
- Required gradient can be computed fast using backprop

Adversarial Training of logistic regression

- Fast gradient method is exact
 - Most damaging example in max norm box

Weights of logistic regression model



Sign of weights of logistic regression model

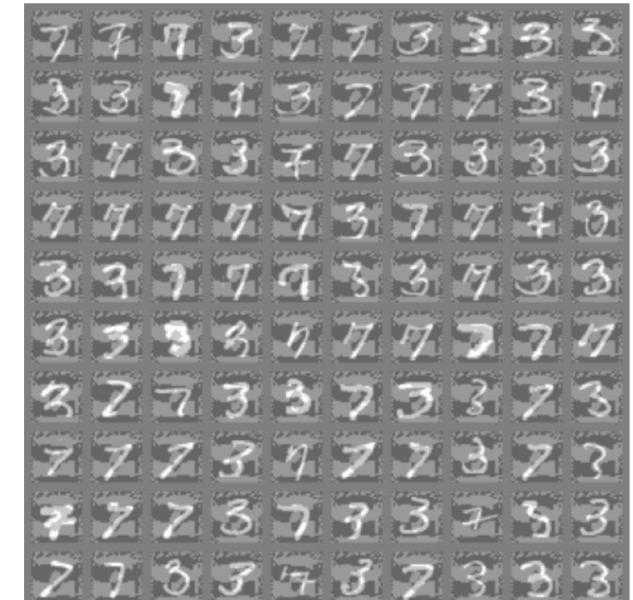


MNIST 3s and 7s

7	7	7	3	7	7	3	3	3
3	3	7	1	3	7	7	7	3
3	7	3	3	7	7	3	3	3
7	7	7	7	7	3	7	7	3
3	3	7	7	7	1	3	7	3
3	3	3	3	7	7	3	3	3
7	7	7	7	7	7	3	7	7
3	3	7	7	7	1	3	7	3
3	3	3	3	7	7	7	7	3

Logistic regression has error rate of 1.6% in 3 vs 7

FGSM adversarial examples with $\epsilon = .25$

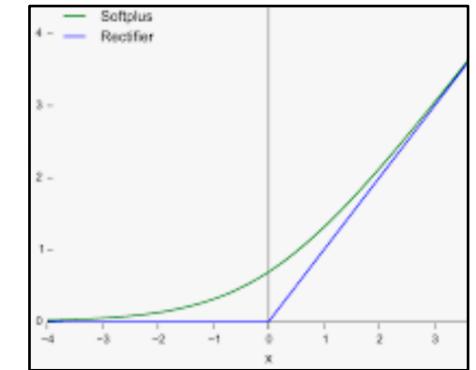


Logistic regression has error rate of 99% in 3 vs 7

Logistic Regression FGSM

- Train to recognize $y \in \{-1, 1\}$ with $P(y = 1) = \sigma(\mathbf{w}^T \mathbf{x} + b)$
 - Training consists of gradient descent on

$$E_{x,y \sim p_{\text{data}}} \zeta(-y(\mathbf{w}^T \mathbf{x} + b))$$
 - where $\zeta(z) = \log(1 + \exp(z))$ is softplus
 - Derivative of softplus is sigmoid
 - Sign of the gradient is $-\text{sign}(\mathbf{w})$, and $\mathbf{w}^T \text{sign}(\mathbf{w}) = \|\mathbf{w}\|_1$
- The adversarial version of logistic regression is therefore to minimize $E_{x,y \sim p_{\text{data}}} \zeta(-y \in \|\mathbf{w}\|_1 - \mathbf{w}^T \mathbf{x} - b)$
 - Unlike L^1 regularization, the L^1 penalty is subtracted off the model's activation during training



L^1 weight decay vs FGSM

- The penalty can eventually start to disappear if the model learns to make confident enough predictions that ζ saturates
- This is not guaranteed to happen
 - In the underfitting regime, adversarial training will simply worsen underfitting
- We can thus view L^1 weight decay as being more “worst case” than adversarial training, because it fails to deactivate in the case of good margin.

Adversarial Training of Deep Nets

- Training on a mixture of adversarial and clean examples
- Training is different from data augmentation
 - Usual augmentation is data with transformations that are expected to actually occur in data
 - This form uses inputs unlikely to occur naturally
 - But expose flaws in the ways that the model conceptualizes its decision function
- Adversarial objective function based on FSG:

$$\tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1-\alpha) J(\theta, x + \varepsilon \text{ sign}(\nabla_x J(\theta, x, y)))$$

Uses of adversarial training

- Adversarial examples have many implications
 - E.g., they are useful in computer security
 - Adversarial examples are hard to defend against
 - They are interesting in the context of regularization
 - Using adversarially perturbed samples we can reduce error rate on test set

Adversarial Training

- Adversarial training discourages highly sensitive local behavior
- By encouraging network to be locally constant in the neighborhood of the training data
- This can be seen as a way of explicitly introducing a local constancy prior into supervised neural nets

Adversarial training and Capacity

- Adversarial training illustrates the power of using a large function family in combination with aggressive regularization
 - Purely linear models, like logistic regression, are unable to resist adversarial examples because they are forced to be linear
- Neural networks are able to represent functions that can range from nearly linear to nearly locally constant
 - Thus can capture linear trends as well as learning to resist local perturbation

Relation to Semi-supervised Learning

- Adversarial examples provide a means of accomplishing semi-supervised learning
- At a point x that is not associated with a label in a dataset, the model itself assigns some label \hat{y}
- It may not be the true label, but if model is of high quality then \hat{y} has a probability of being the true label
- We can seek an adversarial example x' that causes the classifier to output a label y' with $y' \neq \hat{y}$

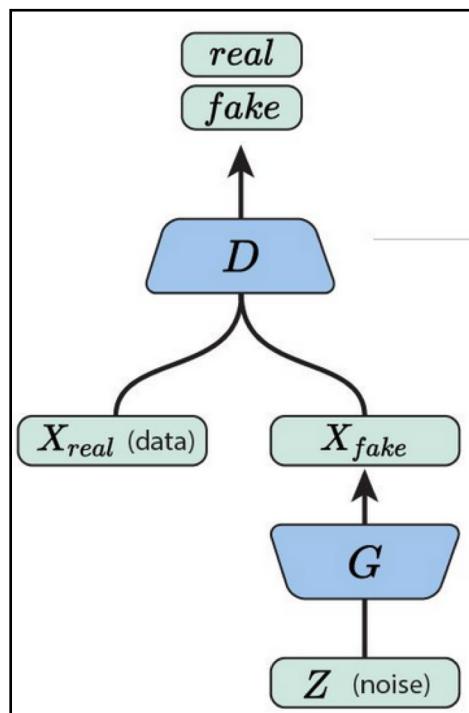
Virtual Adversarial Examples

- Adversarial examples generated with using not the true label but a label provided by a trained model are called *Virtual Adversarial Examples*
 - The classifier may then be trained to assign the same label to x and x'
 - This encourages the classifier to learn a function that is robust to small changes anywhere along the manifold where the unlabeled data lie
- Assumption motivating this approach
 - different classes lie on disconnected manifolds
 - A small perturbation should not be able to jump from one class manifold to another class manifold

Adversarial Training is different from GAN

- Goal of adversarial training: to develop better discriminative models
- Goal of a GAN: to obtain a generative model
 - By having two networks compete with each other

GAN design



The discriminator tries to distinguish genuine data from forgeries created by the generator

The generator turns random noise into imitations of the data, in an attempt to fool the discriminator