Adversarial Examples and Adversarial Training

Ian Goodfellow, OpenAI Research Scientist
NIPS 2016 Workshop on Reliable ML in the Wild
2016-12-9
In this presentation

• “Intriguing Properties of Neural Networks” Szegedy et al, 2013

• “Explaining and Harnessing Adversarial Examples” Goodfellow et al 2014

• “Adversarial Perturbations of Deep Neural Networks” Warde-Farley and Goodfellow, 2016
In this presentation

• “Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples” Papernot et al 2016

• “Practical Black-Box Attacks against Deep Learning Systems using Adversarial Examples” Papernot et al 2016

• “Adversarial Perturbations Against Deep Neural Networks for Malware Classification” Grosse et al 2016

(not my own work)
In this presentation


- Also be sure to check out Takeru Miyato et al’s work on *virtual* adversarial training.
Overview

• What are adversarial examples?

• Why do they happen?

• How can they be used to compromise machine learning systems?

• What are the defenses?

• How to use adversarial examples to improve machine learning, even when there is no adversary
Adversarial Examples

Timeline:
“Adversarial Classification” Dalvi et al 2004: fool spam filter
“Evasion Attacks Against Machine Learning at Test Time” Biggio 2013: fool neural nets
Szegedy et al 2013: fool ImageNet classifiers imperceptibly
Goodfellow et al 2014: cheap, closed form attack
Turning Objects into “Airplanes”
Attacking a Linear Model
Not just for neural nets

- Linear models
  - Logistic regression
  - Softmax regression
  - SVMs
- Decision trees
- Nearest neighbors
Adversarial Examples from Overfitting
Adversarial Examples from Excessive Linearity
Modern deep nets are very piecewise linear

Rectified linear unit

Maxout

Carefully tuned sigmoid

LSTM

(Goodfellow 2016)
Nearly Linear Responses in Practice

(Goodfellow 2016)
Small inter-class distances

All three perturbations have L2 norm 3.96
This is actually small. We typically use 7!
The Fast Gradient Sign Method

\[ J(\tilde{x}, \theta) \approx J(x, \theta) + (\tilde{x} - x)\top \nabla_x J(x). \]

Maximize

\[ J(x, \theta) + (\tilde{x} - x)\top \nabla_x J(x) \]

subject to

\[ \|\tilde{x} - x\|_{\infty} \leq \epsilon \]

\[ \Rightarrow \tilde{x} = x + \epsilon \text{sign} (\nabla_x J(x)) . \]
Maps of Adversarial and Random Cross-Sections

(collaboration with David Warde-Farley and Nicolas Papernot)
Clever Hans

(“Clever Hans, Clever Algorithms,” Bob Sturm)
Wrong almost everywhere
RBFs behave more intuitively
Cross-model, cross-dataset, cross-technique generalization
Transferability Attack

Target model with unknown weights, machine learning algorithm, training set; maybe non-differentiable

Train your own model

Substitute model mimicking target model with known, differentiable function

Adversarial crafting against substitute

Deploy adversarial examples against the target; transferability property results in them succeeding

(Goodfellow 2016)
Practical Attacks

• Fool real classifiers trained by remotely hosted API (MetaMind, Amazon, Google)

• Fool malware detector networks

• Display adversarial examples in the physical world and fool machine learning systems that perceive them through a camera
Adversarial Examples in the Physical World

(a) Printout

(b) Photo of printout

(c) Cropped image

(Goodfellow 2016)
Failed defenses

Generative pretraining

Adding noise at test time

Confidence-reducing perturbation at test time

Weight decay

Various non-linear units

Removing perturbation with an autoencoder

Ensembles

Error correcting codes

Multiple glimpses

Double backprop

Dropout

Adding noise at train time

(Goodfellow 2016)
Training on Adversarial Examples

- Train=Clean, Test=Clean
- Train=Clean, Test=Adv
- Train=Adv, Test=Clean
- Train=Adv, Test=Adv

Test misclassification rate vs Training time (epochs)

(Goodfellow 2016)
Universal engineering machine (model-based optimization)

Make new inventions by finding input that maximizes model’s predicted performance

Training data  Extrapolation

(Goodfellow 2016)
Conclusion

- Attacking is easy
- Defending is difficult
- Benchmarking vulnerability is training
- Adversarial training provides regularization and semi-supervised learning
- The out-of-domain input problem is a bottleneck for model-based optimization generally
cleverhans

Open-source library available at:
https://github.com/openai/cleverhans
Built on top of TensorFlow (Theano support anticipated)
Standard implementation of attacks, for adversarial training and reproducible benchmarks