Fooling Neural Networks

Linguang Zhang
Feb-4-2015
Preparation

• Task: image classification.

• Datasets: MNIST, ImageNet.

• training and testing data.
• Logistic regression:

• Good for 0/1 classification. e.g. spam filtering

$$h_\theta(x) = \frac{1}{1 + \exp(-\theta^T x)}.$$  

$$J(\theta) = -\frac{1}{m} \left[ \sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)})) \right]$$
Preparation

• Multi-class classification? N categories?

• Softmax regression

• Weight Decay (regularization)

\[ h_\theta(x^{(i)}) = \begin{bmatrix} p(y^{(i)} = 1|x^{(i)}; \theta) \\ p(y^{(i)} = 2|x^{(i)}; \theta) \\ \vdots \\ p(y^{(i)} = k|x^{(i)}; \theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{k} e^{\theta_j^T x^{(i)}}} \begin{bmatrix} e^{\theta_1^T x^{(i)}} \\ e^{\theta_2^T x^{(i)}} \\ \vdots \\ e^{\theta_k^T x^{(i)}} \end{bmatrix} \]

\[ J(\theta) = -\frac{1}{m} \left[ \sum_{i=1}^{m} \sum_{j=1}^{k} 1 \{y^{(i)} = j\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}} \right] + \frac{\lambda}{2} \sum_{i=1}^{k} \sum_{j=0}^{n} \theta_{ij}^2 \]
Preparation

• Autoencoder

• What is autoencoder?

\[\text{Input} = \text{decoder}(\text{encoder}(\text{input}))\]

• Why is it useful? Dimension reduction.

• Training

  • Feed-forward and obtain output \(\hat{x}\) at the output layer
  
  • Compute dist(\(\hat{x}, x\)).
  
  • Update weights through backpropagation.
Basic Neural Network
Intriguing Properties of Neural Networks

Activation $\phi(x)$

- Activation of a hidden unit is a meaningful feature.

using the natural basis of the i-th hidden unit:

$$x' = \arg \max_{x \in \text{image set}} \langle \phi(x), e_i \rangle$$

randomly choose a vector:

$$x' = \arg \max_{x \in \text{image set}} \langle \phi(x), v \rangle$$
using the natural basis:

(b) Unit sensitive to upper round stroke, or lower straight stroke.

d) Unit sensitive to diagonal straight stroke.

randomly choose a vector:

(b) Direction sensitive to lower left loop.

d) Direction sensitive to right, upper round stroke.
Adversarial Examples

• What is adversarial example? We can let the network to misclassify an image by adding a imperceptible (for human) perturbation.

• Why do adversarial examples exist? Deep Neural Networks learn input-output mappings that are discontinuous to a significant extent.

• Interesting observation: the adversarial examples generated for network A can also make network B fail.
Generate Adversarial Examples

Input image: \( x \in \mathbb{R}^m \)

Classifier: \( f : \mathbb{R}^m \rightarrow \{1 \ldots k\} \)

Target label: \( l \in \{1 \ldots k\} \)

Minimize \( \|r\|_2 \) subject to:

1. \( f(x + r) = l \)
2. \( x + r \in [0, 1]^m \)

When \( f(x) \neq l \):

Minimize \( cr + \text{loss}_f(x + r, l) \) subject to \( x + r \in [0, 1]^m \)

\( x + r \) is the closest image to \( x \) classified as \( l \) by \( f \).
Intriguing properties

• Properties:
  • Visually hard to distinguish the generated adversarial examples.
  • Cross model generalization. (different hyper-parameters)
  • Cross training-set generalization. (different training set)

• Observation:
  • adversarial examples are universal.
  • back-feeding adversarial examples to training might improve generalization of the model.
Experiment

Cross-model generalization of adversarial examples.

<table>
<thead>
<tr>
<th></th>
<th>softmax1</th>
<th>softmax2</th>
<th>softmax3</th>
<th>N100-100-10</th>
<th>N200-200-10</th>
<th>AE400-10</th>
<th>Av. distortion</th>
</tr>
</thead>
<tbody>
<tr>
<td>softmax with $\lambda = 10^{-4}$</td>
<td>100%</td>
<td>11.7%</td>
<td>22.7%</td>
<td>2%</td>
<td>3.9%</td>
<td>2.7%</td>
<td>0.062</td>
</tr>
<tr>
<td>softmax with $\lambda = 10^{-2}$</td>
<td>87.1%</td>
<td>100%</td>
<td>35.2%</td>
<td>35.9%</td>
<td>27.3%</td>
<td>9.8%</td>
<td>0.1</td>
</tr>
<tr>
<td>softmax with $\lambda = 1$</td>
<td>71.9%</td>
<td>76.2%</td>
<td>100%</td>
<td>48.1%</td>
<td>47%</td>
<td>34.4%</td>
<td>0.14</td>
</tr>
<tr>
<td>N100-100-10</td>
<td>28.9%</td>
<td>13.7%</td>
<td>21.1%</td>
<td>100%</td>
<td>6.6%</td>
<td>2%</td>
<td>0.058</td>
</tr>
<tr>
<td>N200-200-10</td>
<td>38.2%</td>
<td>14%</td>
<td>23.8%</td>
<td>20.3%</td>
<td>100%</td>
<td>2.7%</td>
<td>0.065</td>
</tr>
<tr>
<td>AE400-10</td>
<td>23.4%</td>
<td>16%</td>
<td>24.8%</td>
<td>9.4%</td>
<td>6.6%</td>
<td>100%</td>
<td>0.086</td>
</tr>
<tr>
<td>Gaussian noise, stddev=0.1</td>
<td>5.0%</td>
<td>10.1%</td>
<td>18.3%</td>
<td>0%</td>
<td>0%</td>
<td>0.8%</td>
<td>0.1</td>
</tr>
<tr>
<td>Gaussian noise, stddev=0.3</td>
<td>15.6%</td>
<td>11.3%</td>
<td>22.7%</td>
<td>5%</td>
<td>4.3%</td>
<td>3.1%</td>
<td>0.3</td>
</tr>
</tbody>
</table>
## Experiment

Cross training-set generalization - baseline (no distortion)

<table>
<thead>
<tr>
<th>Model</th>
<th>Error on $P_1$</th>
<th>Error on $P_2$</th>
<th>Error on Test</th>
<th>Min Av. Distortion</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_1$: 100-100-10 trained on $P_1$</td>
<td>0%</td>
<td>2.4%</td>
<td>2%</td>
<td>0.062</td>
</tr>
<tr>
<td>$M'_1$: 123-456-10 trained on $P_1$</td>
<td>0%</td>
<td>2.5%</td>
<td>2.1%</td>
<td>0.059</td>
</tr>
<tr>
<td>$M_2$: 100-100-10 trained on $P_2$</td>
<td>2.3%</td>
<td>0%</td>
<td>2.1%</td>
<td>0.058</td>
</tr>
</tbody>
</table>

Cross training-set generalization error rate

<table>
<thead>
<tr>
<th>Distortion Type</th>
<th>$M_1$</th>
<th>$M'_1$</th>
<th>$M_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distorted for $M_1$ (av. stddev=0.062)</td>
<td>100%</td>
<td>26.2%</td>
<td>5.9%</td>
</tr>
<tr>
<td>Distorted for $M'_1$ (av. stddev=0.059)</td>
<td>6.25%</td>
<td>100%</td>
<td>5.1%</td>
</tr>
<tr>
<td>Distorted for $M_2$ (av. stddev=0.058)</td>
<td>8.2%</td>
<td>8.2%</td>
<td>100%</td>
</tr>
<tr>
<td>Gaussian noise with stddev=0.06</td>
<td>2.2%</td>
<td>2.6%</td>
<td>2.4%</td>
</tr>
<tr>
<td>Distorted for $M_1$ amplified to stddev=0.1</td>
<td>100%</td>
<td>98%</td>
<td>43%</td>
</tr>
<tr>
<td>Distorted for $M'_1$ amplified to stddev=0.1</td>
<td>96%</td>
<td>100%</td>
<td>22%</td>
</tr>
<tr>
<td>Distorted for $M_2$ amplified to stddev=0.1</td>
<td>27%</td>
<td>50%</td>
<td>100%</td>
</tr>
<tr>
<td>Gaussian noise with stddev=0.1</td>
<td>2.6%</td>
<td>2.8%</td>
<td>2.7%</td>
</tr>
</tbody>
</table>

*Note: magnify distortion*
The Opposite Direction

Imperceptible adversarial examples that cause misclassifcation.

Unrecognizable images that make DNN believe

Fooling Examples

Problem statement: producing images that are completely unrecognizable to humans, but that state-of-the-art Deep Neural Networks believe to be recognizable objects with high confidence (99%).
DNN Models

- ImageNet: AlexNet. (Caffe version)
  - 42.6% error rate. Original error rate is 40.7%.
- MNIST: LeNet (Caffe version)
  - 0.94% error rate. Original error rate is 0.8%.
Generating Images with Evolution (one class)

- Evolutionary Algorithms (EAs) are inspired by Darwinian evolution.
- Contains a population of organisms (images).
- Organisms will be randomly perturbed and selected based on fitness function.
- Fitness function: in our case, is the highest prediction value a DNN believes that the image belongs to a class.
Generating Images with Evolution (multi-class)

- Algorithm: Multi-dimensional archive of phenotypic elites MAP-Elites.

- Procedures:
  - Randomly choose an organism, mutate it randomly.
  - Show the mutated organism to the DNN. If the prediction score is higher than the current highest score of **ANY** class, make the organism as the champion of that class.
Encoding an Image

• Direct encoding:
  
  • For MNIST: 28 x 28 pixels.
  
  • For ImageNet: 256 x 256 pixels, each pixel has 3 channels (H, S, V).
  
  • Values are independently mutated.
    
    • 10% chance of being chosen. The chance drops by half every 1000 generations.
    
    • mutate via the polynomial mutation operator.
Directly Encoded Images

<table>
<thead>
<tr>
<th>robin</th>
<th>cheetah</th>
<th>armadillo</th>
<th>lesser panda</th>
</tr>
</thead>
<tbody>
<tr>
<td>centipede</td>
<td>peacock</td>
<td>jackfruit</td>
<td>bubble</td>
</tr>
</tbody>
</table>
Encoding an Image

• Indirect encoding:
  • very likely to produce regular images with meaningful patterns.
  • both humans and DNNs can recognize.
  • Compositional pattern-producing network (CPPN).
CPPN-encoded Images

- Parrot
- Toucan
- Ocarina
- Pinwheel
- Coffee pot
- Cup
- Coffee mug
- Water jug
- Race car
- Racer
- Sports car
- Car wheel
- Sea
- Seashore
- Pier
- Sandbar
- Black panther
- Mask
- Mouse
- Loupe
- Fly
- Ground beetle
- Fly
- Rhinoceros beetle
- Red crayon
- Syringe
- Lipstick
- Maraca
- Latte
- Cup
- CD player
- Stethoscope
MNIST - Irregular Images

LeNet: 99.99% median confidence, 200 generations.
MNIST - Regular Images

LeNet: 99.99% median confidence, 200 generations.
ImageNet - Irregular Images

AlexNet: 21.59% median confidence, 20000 generations.
45 classes: > 99% confidence.
ImageNet - Irregular Images

<table>
<thead>
<tr>
<th>robin</th>
<th>cheetah</th>
<th>armadillo</th>
<th>lesser panda</th>
</tr>
</thead>
<tbody>
<tr>
<td>centipede</td>
<td>peacock</td>
<td>jackfruit</td>
<td>bubble</td>
</tr>
</tbody>
</table>
AlexNet: 88.11% median confidence, 5000 generations. High confidence images are found in most classes.

Dogs and cats
Difficulties in Dogs and Cats

• Size of dataset of cats and dogs is large.
  • Less overfit -> difficult to fool.

• Too many classes for cats and dogs.
  • e.g. difficult to achieve high score in Dog A while guaranteeing low score in Dog B.

• [Recall] For the final softmax layer, it is difficult to give high confidence in the above case.
ImageNet - Regular Images

- king penguin
- starfish
- baseball
- electric guitar
- freight car
- remote control
- peacock
- African grey
Fooling Closely Related Classes

Lizard classes
- agama: 62.11 %
- frilled lizard: 13.28 %
- green lizard: 90.04 %

Toy dog classes
- Japanese spaniel: 53.18 %
- Pekinese: 63.06 %
- Blenheim spaniel: 44.50 %

Run 1
- agama
- frilled lizard
- green lizard

Run 2
- agama: 92.25 %
- frilled lizard: 85.76 %
- green lizard: 93.29 %

- Japanese spaniel: 50.25 %
- Pekinese: 85.88 %
- Blenheim spaniel: 65.76 %
Fooling Closely Related Classes

• Two possibilities:
  
  • [Recall] Imperceptible changes can change a DNN’s class label. Evolution could produce very similar images to fool multiple classes.
  
  • Many of the images are related to each other naturally.
  
  • Different runs produce different images: many ways to fool the DNN.
Repetition of Patterns

Before

Lipstick: 99.94%
Ruler: 99.90%
Oboe: 99.91%

After

Lipstick: 99.80%
Ruler: 99.40%
Oboe: 99.89%
Repetition of Patterns

• Explanations

  • Extra copies make the DNN more confident.

  • DNNs tend to learn low&mid-level features rather than the global structure.

  • Many natural images do contain multiple copies.
Training with Fooling Images

<table>
<thead>
<tr>
<th>$i$</th>
<th>Error</th>
<th>MNIST Error</th>
<th>Train</th>
<th>Val</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.94</td>
<td>0.94</td>
<td>60000</td>
<td>10000</td>
<td>99.99</td>
</tr>
<tr>
<td>2</td>
<td>1.02</td>
<td>0.87</td>
<td>66000</td>
<td>11000</td>
<td>97.42</td>
</tr>
<tr>
<td>3</td>
<td>0.92</td>
<td>0.87</td>
<td>67000</td>
<td>11100</td>
<td>99.83</td>
</tr>
<tr>
<td>4</td>
<td>0.89</td>
<td>0.83</td>
<td>68000</td>
<td>11200</td>
<td>72.52</td>
</tr>
<tr>
<td>5</td>
<td>0.90</td>
<td>0.96</td>
<td>69000</td>
<td>11300</td>
<td>97.55</td>
</tr>
<tr>
<td>6</td>
<td>0.89</td>
<td>0.99</td>
<td>70000</td>
<td>11400</td>
<td>99.68</td>
</tr>
<tr>
<td>7</td>
<td>0.86</td>
<td>0.98</td>
<td>71000</td>
<td>11500</td>
<td>76.13</td>
</tr>
<tr>
<td>8</td>
<td>0.91</td>
<td>1.01</td>
<td>72000</td>
<td>11600</td>
<td>99.96</td>
</tr>
<tr>
<td>9</td>
<td>0.90</td>
<td>0.86</td>
<td>73000</td>
<td>11700</td>
<td>99.51</td>
</tr>
<tr>
<td>10</td>
<td>0.84</td>
<td>0.94</td>
<td>74000</td>
<td>11800</td>
<td>99.48</td>
</tr>
<tr>
<td>11</td>
<td>0.80</td>
<td>0.93</td>
<td>75000</td>
<td>11900</td>
<td>98.62</td>
</tr>
<tr>
<td>12</td>
<td>0.82</td>
<td>0.98</td>
<td>76000</td>
<td>12000</td>
<td>99.97</td>
</tr>
<tr>
<td>13</td>
<td>0.75</td>
<td>0.90</td>
<td>77000</td>
<td>12100</td>
<td>99.93</td>
</tr>
<tr>
<td>14</td>
<td>0.80</td>
<td>0.96</td>
<td>78000</td>
<td>12200</td>
<td>99.15</td>
</tr>
<tr>
<td>15</td>
<td>0.79</td>
<td>0.95</td>
<td>79000</td>
<td>12300</td>
<td>99.15</td>
</tr>
</tbody>
</table>

Retraining does not help.
adversarial examples

---

Training set

Test set

Adversarial Example [26]

Evolved images, indirectly encoded [this]

Indirect encoding

$G_0$

$I_0$

Direct encoding / gradient method

Image space

Gradient ascent [this, 10, 22] and evolved images, directly encoded [this]
Why Do Adversarial Examples Exist?

- Past explanations
  - extreme nonlinearity of DNN.
  - insufficient model averaging.
  - insufficient regularization.

- New explanation
  - Linear behavior in high-dimensional spaces is sufficient to cause adversarial examples.

Linear Explanations of Adversarial Examples

Perturbation: \( \eta \)

Adversarial examples: \( \tilde{x} = x + \eta \)

Pixel value precision: \( \epsilon \) typically = 1/255

Perturbation is meaningless if: \( ||\eta||_\infty < \epsilon \)

Activation of adversarial examples: \( w^T \tilde{x} = w^T x + w^T \eta \)

\( \eta = \epsilon \text{sign}(w) \) maximizes the increase of activation.
Linear Explanations of Adversarial Examples

Activation of adversarial examples: \( \mathbf{w}^T \tilde{x} = \mathbf{w}^T \mathbf{x} + \mathbf{w}^T \mathbf{\eta} \)

Assume the magnitude of the weight vector is \( \mathbf{m} \) and the dimension is \( \mathbf{n} \):

Increase of activation is: \( \epsilon \mathbf{m} \mathbf{n} \)

A simple linear model can have adversarial examples as long as its input has sufficient dimensionality.
Faster Way to Generate Adversarial Examples

Cost function: \( J(\theta, x, y) \)

Perturbation: \( \eta = \epsilon \text{sign} (\nabla_x J(\theta, x, y)) \)

\begin{align*}
\text{"panda"} & \quad 57.7\% \text{ confidence} \\
\text{sign}(\nabla_x J(\theta, x, y)) & \quad \text{"nematode"} \\
\quad + 0.007 \times & \quad 8.2\% \text{ confidence} \\
\text{x} & \quad \text{x +} \\
\text{99.3 \% confidence} & \quad \epsilon \text{sign}(\nabla_x J(\theta, x, y))
\end{align*}
Faster Way to Generate Adversarial Examples

<table>
<thead>
<tr>
<th></th>
<th>epsilon</th>
<th>error rate</th>
<th>confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>shallow softmax (MNIST)</td>
<td>0.25</td>
<td>99.9%</td>
<td>79.3%</td>
</tr>
<tr>
<td>maxout network</td>
<td>0.25</td>
<td>89.4%</td>
<td>97.6%</td>
</tr>
<tr>
<td>convolutional maxout network (CIFAR-10)</td>
<td>0.1</td>
<td>87.15%</td>
<td>96.6%</td>
</tr>
</tbody>
</table>
Adversarial Training of Linear Models

Simple case: Linear Regression.

\[ P(y = 1 \mid x) = \sigma (w^\top x + b) \]

Train gradient descend on:

\[ \mathbb{E}_{x, y \sim p_{\text{data}}} \zeta(-y(w^\top x + b)) \]

Adversarial training version is:

\[ \mathbb{E}_{x, y \sim p_{\text{data}}} \zeta(y(\epsilon||w||_1 - w^\top x - b)) \]
Adversarial Training of Deep Networks

Regularized cost function:

\[ \tilde{J}(\theta, x, y) = \alpha J(\theta, x, y) + (1 - \alpha) J(\theta, x + \epsilon \text{sign} (\nabla_x J(\theta, x, y))) \]

On MNIST: error rate drops from 0.94% to 0.84%

For adversarial examples: error rate drops from 89.4% to 17.9%

Original model: 40.9%

Adversarially trained model: 19.4%
Explaining Why Adversarial Examples Generalize

• [Recall] An adversarial example generated for one model is often misclassified by other models.

• When different models misclassify an adversarial examples, they often agree with each other.

• As long as $w^T \eta$ is positive, adversarial examples work.

• Hypothesis: neural networks trained all resemble the linear classifier learned on the same training set.

  • Such stability of underlying classification weights causes the stability of adversarial examples.
Fooling Examples

• Can simply generate fooling examples by generating a point far from the data with larger norms (more confidence)

• Gaussian fooling examples:
  • softmax top layer: error rate: 98.35%, average confidence: 92.8%.
  • independent sigmoid top layer: error rate: 68%, average confidence: 87.9%.
Summary

• **Intriguing properties**
  - No difference between individual high level units and random linear combinations of high level units.
  - Adversarial Examples
    - Indistinguishable.
    - Generalize.

• **Fooling images**
  - Generate fooling images via evolution.
    - Direct encoding and indirect encoding (irregular and regular images).
  - Retraining does not boost immunity.
Generative Adversarial Nets

- Two types of models:
  - Generative model: generative model learns the joint probability distribution of the data - $p(x, y)$.
  - Discriminative model: discriminative model learns the conditional probability distribution of the data - $p(y | x)$.
  - Much easier to get discriminative model with the generative model.
Main Idea

- Adversarial process:
  - simultaneously train two models
    - a generative model G captures the data distribution.
    - discriminative model D - tells whether a sample comes from the training data or not.
- Optimal solution:
  - G recovers the data distribution.
  - D is 1/2 everywhere.
Two-player minmax game

\[
\min_{G} \max_{D} V(D, G) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_{z}(z)}[\log(1 - D(G(z)))]
\]
Thanks.