Modelling adversarial training

Michiel Straat
• Adversarial examples,

• Compacting adversarial

• Logging adversarial

• Validating adversarial

• Assembling adversarial
Adversarial examples

Resnet50 trained in imagenet: Kit fox with 76% confidence
Kit fox with 76.0% confidence

Max: 2/255, min: -2/255

Coyote with 99.8% confidence.
Kit fox with 2.7*10^-6% confidence.
Constructing adversarial examples

Adversarial example: tiny perturbations applied to data aimed at causing incorrect predictions.

For an example \( \xi^{(\mu)} \in \mathbb{R}^N \) find small \( \delta^{(\mu)} \in \Delta \) such that \( \arg\max_i \sigma(\xi + \delta)_i \neq \tau^{(\mu)} \)

The perturbations should be imperceptible or not change the semantics of the data:

\[
\Delta = \{ \delta : \|\delta\|_\infty \leq \epsilon \}
\]

(each component of \( \xi \) perturbable at most by \([-\epsilon, \epsilon]\))

Apply a perturbation \( \delta \) that maximizes the loss:

\[
\max_{\delta \in \Delta} \mathcal{L}(\sigma_{\mathbf{w}}(\xi + \delta), \tau)
\]

In practice, approximate above by:

\[
\tilde{\xi} = \xi + \epsilon \cdot \text{sign}(\nabla \xi \mathcal{L}(\sigma(\xi), \tau))
\]
Targeted attack

$$\max_{\delta \in \Delta} \left( \sigma_w(\xi + \delta)_{\tau_{\text{goal}}} - \sigma_w(\xi + \delta)_\tau \right)$$

Max: 2/255, min: -2/255

Kit fox with 76.0% confidence

+  

Whisky jug with 100.0% confidence
Computing adversarial perturbations

Fast Gradient Sign Method:

\[
\max_{\delta \in \Delta} \mathcal{L}(\mathbf{w}(\xi + \delta), \tau)
\]
\[
\Delta = \{ \delta : \|\delta\|_\infty \leq \epsilon \}
\]

\[
\tilde{\xi} = \xi + \epsilon \cdot \text{sign}(\nabla_{\xi} \mathcal{L}(\sigma(\xi), \tau))
\]

Basic Iterative Method:

\[
\xi^{(0)} = \xi
\]

\[
\xi^{(t)} = \text{proj}(\xi^{(t-1)} + \epsilon \cdot \text{sign}(\nabla_{\xi} \mathcal{L}(\sigma(\xi^{(t)}), \tau)))
\]

For \(t=[1,2,...,T]\) iterations

- If only discrete labels available:
  train surrogate model on data labeled by the target model and compute adversarial examples on that model (using above), (same data distr \(\rightarrow\) similar models).
Adversarial training

Assume data, $(\xi, \tau) \in \mathbb{R}^N \times [1, 2, \ldots K]$ with a distribution $P$.

$\sigma : \mathbb{R}^N \rightarrow [0, 1]^K$ is a classifier.

Standard training:

$$\min_w \mathbb{E}_{(\xi, \tau) \sim P}(\mathcal{L}(\sigma_w(\xi), \tau))$$

Adversarial training: allow an adversary to perturb the input first:

$$\min_w \max_{\delta \in \Delta} \mathbb{E}_{(\xi, \tau) \sim P}(\mathcal{L}(\sigma_w(\xi + \delta), \tau))$$

“Towards Deep Learning Models Resistant to Adversarial Attacks”, A. Madry et al, ICLR 2018
Assessing robustness of the model

- The standard classification loss of the classifier is
  \[ L(\sigma) = \mathbb{E}_{(\xi, \tau) \sim P} [\arg \max_i \sigma(\xi)_i \neq \tau] \]

- The \( \epsilon \)-adversarial classification loss counts considers in addition the non-robust examples:
  \[ L_\epsilon(\sigma) = \mathbb{E}_{(\xi, \tau) \sim P} [\arg \max_i [\sigma(\xi + \delta)_i \neq \tau] \]

- By varying \( \epsilon \) in the above definition, one obtains a *monotonically increasing* adversarial loss curve.

"Adversarial robustness curves", C. Göpfert, J. Göpfert, B. Hammer, ECML/PKDD 2019
Experiment Logistic Regression MNIST

\[
\sigma(\xi) = \zeta(\mathbf{w} \cdot \xi + b)
\]

Class labels \( \tau = [-1, 1] \)

1: label 3.
-1: label is 7.

"Explaining and Harnessing Adversarial Examples", I. Goodfellow et al
Applying the adversarial perturbation to all test samples: 40% test error.
Experiment Logistic Regression MNIST

Test error for increasing perturbation strength

"Explaining and Harnessing Adversarial Examples", I. Goodfellow et al
Modelling of adversarial training: 1) the data distribution

Stream of examples $\xi \in \mathbb{R}^N$

$$P(\xi) = p \mathcal{N}(\xi | \mathbf{B}_1, v_1 I) + (1 - p) \mathcal{N}(\xi | \mathbf{B}_2, v_2 I)$$

$\mathbf{B} \in \mathbb{R}^N$, $||\mathbf{B}||_2 = 1$

Means of the two modes:

$$\mathbf{B}_1 = \frac{1}{2} \rho \mathbf{B}$$

$$\mathbf{B}_2 = -\frac{1}{2} \rho \mathbf{B} = -\mathbf{B}_1$$

(Separation between cluster centers is $\rho$)

Labels:

$$\tau^{(\mu)} = \begin{cases} 
1 & \text{if } \xi^\mu \text{ from } \mathcal{N}(\xi | \mathbf{B}_1, v_1 I) \\
-1 & \text{if } \xi^\mu \text{ from } \mathcal{N}(\xi | \mathbf{B}_2, v_2 I)
\end{cases}$$
1) the data distribution, data in high dimension $N=1000$

$$N = 1000, \rho = 6.0, v_1 = v_2 = 1$$
Modelling of adversarial training: 2) the machine learning model

Model with linear decision boundary, parameters are weight vector and bias:

\[ \mathbf{w} \in \mathbb{R}^N, \ b \in \mathbb{R} \]

\[ \| \mathbf{w} \|_2 = 1 \]

Then the output of the model is

\[ \sigma(\xi^{(\mu)}) = \text{erf}(\mathbf{w} \cdot \xi^{(\mu)} - b) \]

Define the argument as:

\[ x^{(\mu)} = \mathbf{w} \cdot \xi^{(\mu)} - b \]

(Local potential/pre-activation)
Modelling of adversarial training: 3) the learning algorithm

<table>
<thead>
<tr>
<th>Normal algorithm</th>
<th>Adversarial Training (AT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training data</td>
<td>$\xi^{(\mu)}$</td>
</tr>
</tbody>
</table>

Loss:

$$\mathcal{L}(\{w, b\}, \tilde{\xi}^{(\mu)}, \tau^{(\mu)}) = \frac{1}{2}(\tilde{\sigma}^{(\mu)} - \tau^{(\mu)})^2$$

Updates:

$$w^{(\mu+1)} = (w^{(\mu)} - \frac{\eta_w}{N} \nabla_w \mathcal{L}) / \|\cdot\|$$

$$b^{(\mu+1)} = b^{(\mu)} - \frac{\eta_b}{N} \nabla_b \mathcal{L}$$
Modelling of adversarial training: Order parameters

\[
\tilde{\xi}(\mu) = \xi(\mu) - \tau(\mu) \epsilon w
\]
\[
\tilde{\sigma}(\mu) = \text{erf}(\tilde{\xi}(\mu))
\]
with pre-activation: \( \tilde{x}(\mu) = w \cdot \tilde{\xi}(\mu) - b = w \cdot \xi(\mu) - \tau(\mu) \epsilon - b \)

\( \tilde{\nu} \) is distributed as a Gaussian (in this case also for small N)

It's conditional mean and variance are:

\[
\langle \tilde{x} \rangle_1 = \frac{1}{2} \rho \left( w \cdot B \right) - b - \epsilon, \quad \text{Var} [\tilde{x}]_1 = \nu_1
\]
\[
\langle \tilde{x} \rangle_2 = -\frac{1}{2} \rho R - b + \epsilon, \quad \text{Var} [\tilde{x}]_2 = \nu_2
\]

R and b are order parameters of the system.
Modelling of adversarial training: Dynamics of the order parameters

\[ R = w \cdot B \]

\[
\mathbf{w}^{(\mu+1)} = (w^{(\mu)} - \frac{\eta_{w}}{N} \nabla_w \mathcal{L})/|| \cdot ||
\]

\[
b^{(\mu+1)} = b^{(\mu)} - \frac{\eta_{b}}{N} \nabla_b \mathcal{L}
\]

\[
R^{(\mu+1)} - R^{(\mu)} = \frac{\eta_{w}}{N} f(\tilde{x}, \tilde{y})
\]

\[
b^{(\mu+1)} - b^{(\mu)} = \frac{\eta_{b}}{N} h(\tilde{x})
\]

Self-averages for large N and many example presentations for the normalized learning time \( \alpha = \mu/N \), write ODEs:

\[
\frac{\partial R}{\partial \alpha} = \eta_{w} \langle f(\tilde{x}, \tilde{y}) \rangle P(\tilde{x}, \tilde{y})
\]

\[
\frac{\partial b}{\partial \alpha} = \eta_{b} \langle h(\tilde{x}) \rangle P(\tilde{x})
\]
Modelling of adversarial training: Evaluation measures

Test classification error:

$$\mathcal{E}(R, b) = p\langle \Theta(-x) \rangle_{P_1(x)} + (1 - p)\langle \Theta(x) \rangle_{P_2(x)}$$

Adversarial classification error:

$$S(R, b, \epsilon) = p\langle \Theta(\epsilon - x) \rangle_{P_1(x)} + (1 - p)\langle \Theta(x + \epsilon) \rangle_{P_2(x)}$$

Obtain both curves by substitution of $R(\alpha), b(\alpha)$:

$$\mathcal{E}(R(\alpha), b(\alpha)), \ S(R(\alpha), b(\alpha), \epsilon)$$

Obtain final robustness vs. a range of epsilon:

$$S(R(\alpha_{\text{max}}), b(\alpha_{\text{max}}), \epsilon = [0...\epsilon_{\text{max}}])$$
Results – Scenario 1

Data: \( N = 1000, \ \text{prior}=0.5, \ \rho = 2.0, \ v_1 = v_2 = 1.0 \)

Algo: \( \eta_w = 0.04, \ \eta_v = 0.2, \ \text{AT: } \epsilon = 1 \)

**Order parameter \( R \)**

**Bias**

**Test classification error**

\( \epsilon \)-adversarial loss for \( \epsilon = 1 \) during training

\( \epsilon \)-adversarial loss vs. \( \epsilon \)
Results – Scenario 2

Data: $N = 1000$, $\text{prior} = 0.7$, $\rho = 2.0$, $\nu_1 = \nu_2 = 1.0$

Algo: $\eta_w = 0.04$, $\eta_b = 0.2$, AT: $\epsilon = 1$

Order parameter $R$

Bias

Test classification error

$\epsilon$-adversarial loss for $\epsilon = 1$ during training

$\epsilon$-adversarial loss vs. $\epsilon$
Data: $N = 1000$, prior=0.8, $\rho = 3.0$, $v_1 = 1.0$, $v_2 = 3.0$
Algo: $\eta_w = 0.04$, $\eta_b = 0.2$, AT: $\epsilon = 1.0$
Outlook

- Extension to two layer non-linear neural networks, better model scenarios.
- Analyse variants of adversarial training, randomized epsilon, training only on non-robust examples (hinge loss).