Adversarially Regularized Autoencoders

Junbo (Jake) Zhao, Yoon Kim, Kelly Zhang, Alexander M. Rush, Yann LeCun

Wei Zhen Teoh and Mathieu Ravaut
Refresh: Adversarial Autoencoder

[From Adversarial Autoencoders by Makhzani et al 2015]
Some Changes - Learned Generator

Generator distribution is also learned

Adversarial cost for distinguishing positive samples $p(z)$ from negative samples $q(z)$
The distance measure between two distributions is defined by the Earth-mover distance, or Wasserstein-1:

\[
W(P_r, P_g) = \inf_{\gamma \in \Pi(P_r, P_g)} \mathbb{E}_{(x, y) \sim \gamma} \left[ \|x - y\| \right],
\]

where \(\Pi(P_r, P_g)\) denotes the set of all joint distributions \(\gamma(x, y)\) whose marginals are respectively \(P_r\) and \(P_g\).
Some Changes - Wasserstein GAN

- This is equivalent to the following supremum over Lipschitz-1 functions:

\[
W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)]
\]

- In practice, \( f \) is approximated by a neural network \( f_w \) where all the weights are clipped to lie in a compact space such as a hypercube of size \( \text{epsilon} \).
Some Changes - Discrete Data

Instead of a continuous vector, $X$ is now discrete data:

- Binarized MNIST

- Text (sequences of one-hot vocabulary vector)

[From https://ayearofai.com/lenny-2-autoencoders-and-word-embeddings-oh-my-576403b0113a]
Some Changes - Encoder (for sequential data)

$h_n$ becomes the latent code $c$

Model

discrete struct. encoder code ($P_r$) decoder reconstruction loss

$x \sim P_x \xrightarrow{\text{enc}_\phi} c \xrightarrow{p_\psi} \tilde{x}$

$z \sim N \xrightarrow{g_\theta} \tilde{c} \xrightarrow{f_w} W$

latent var. generator code ($P_g$) critic regularization

$L_{rec}+$

$W(P_g, P_r)$
Training Objective

\[ \min_{\phi, \psi, \theta} \mathcal{L}_{\text{rec}}(\phi, \psi) + \lambda^{(1)} W(\mathbb{P}_r, \mathbb{P}_g) \]

- Reconstruction loss
- Wasserstein distance between two distributions
Training Objective Components

- Reconstruction from decoder:
  \[
  \hat{x} = \arg \max_x p_\psi(x | \text{enc}_\phi(x))
  \]

- Reconstruction loss:
  \[
  \mathcal{L}_{\text{rec}}(\phi, \psi) = -\log p_\psi(x | \text{enc}_\phi(x))
  \]
Training Objective Components

Discriminator maximizing objective:
\[ L_{\text{cri}}(w) = \mathbb{E}_{x \sim P_x} [f_w(\text{enc}_\phi(x))] - \mathbb{E}_{\tilde{c} \sim P_g} [f_w(\tilde{c})] \]

Generator minimizing objective:
\[ L_{\text{encs}}(\phi, \theta) = \mathbb{E}_{x \sim P_x} [f_w(\text{enc}_\phi(x))] - \mathbb{E}_{\tilde{c} \sim P_g} [f_w(\tilde{c})] \]

The max of this function approximates the Wasserstein distance
Algorithm 1 ARAE Training

for number of training iterations do

(1) Train the autoencoder for reconstruction $[\mathcal{L}_{\text{rec}}(\phi, \psi)]$.

Sample $\{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_x$ and compute code-vectors $c^{(i)} = \text{enc}_\phi(x^{(i)})$.
Backpropagate reconstruction loss, $\mathcal{L}_{\text{rec}} = -\frac{1}{m} \sum_{i=1}^m \log p_\psi(x^{(i)} | c^{(i)}, [y^{(i)}])$, and update.
Algorithm 1 ARAE Training

for number of training iterations do

(1) **Train the autoencoder for reconstruction** $[\mathcal{L}_{\text{rec}}(\phi, \psi)]$.
   - Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_x$ and compute code-vectors $c^{(i)} = \text{enc}_\phi(x^{(i)})$.
   - Backpropagate reconstruction loss, $\mathcal{L}_{\text{rec}} = -\frac{1}{m} \sum_{i=1}^{m} \log p_\psi(x^{(i)} | c^{(i)}, [y^{(i)}])$, and update.

(2) **Train the critic** $[\mathcal{L}_{\text{cri}}(w)]$ (Repeat k times)
   - Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_x$ and $\{z^{(i)}\}_{i=1}^{m} \sim \mathcal{N}(0, I)$.
   - Compute code-vectors $c^{(i)} = \text{enc}_\phi(x^{(i)})$ and $\tilde{c}^{(i)} = g_\theta(z^{(i)})$.
   - Backpropagate loss $-\frac{1}{m} \sum_{i=1}^{m} f_w(c^{(i)}) + \frac{1}{m} \sum_{i=1}^{m} f_w(\tilde{c}^{(i)})$, update, clip the critic $w$ to $[-\epsilon, \epsilon]^d$. 


Algorithm 1 ARAE Training

for number of training iterations do

(1) Train the autoencoder for reconstruction \([\mathcal{L}_{\text{rec}}(\phi, \psi)]\).
   Sample \(\{x^{(i)}\}_{i=1}^{m} \sim P_x\) and compute code-vectors \(c^{(i)} = \text{enc}_\phi(x^{(i)})\).
   Backpropagate reconstruction loss, \(\mathcal{L}_{\text{rec}} = -\frac{1}{m} \sum_{i=1}^{m} \log p_\psi(x^{(i)} | c^{(i)}, [y^{(i)}])\), and update.

(2) Train the critic \([\mathcal{L}_{\text{cri}}(\theta)]\) (Repeat k times)
   Sample \(\{x^{(i)}\}_{i=1}^{m} \sim P_x\) and \(\{z^{(i)}\}_{i=1}^{m} \sim \mathcal{N}(0, \mathbf{I})\).
   Compute code-vectors \(c^{(i)} = \text{enc}_\phi(x^{(i)})\) and \(\tilde{c}^{(i)} = g_\theta(z^{(i)})\).
   Backpropagate loss \(-\frac{1}{m} \sum_{i=1}^{m} f_w(c^{(i)}) + \frac{1}{m} \sum_{i=1}^{m} f_w(\tilde{c}^{(i)})\), update, clip the critic \(w\) to \([-\epsilon, \epsilon]^d\).

(3) Train the generator and encoder adversarially to critic \([\mathcal{L}_{\text{encs}}(\phi, \theta)]\)
   Sample \(\{x^{(i)}\}_{i=1}^{m} \sim P_x\) and \(\{z^{(i)}\}_{i=1}^{m} \sim \mathcal{N}(0, \mathbf{I})\)
   Compute code-vectors \(c^{(i)} = \text{enc}_\phi(x^{(i)})\) and \(\tilde{c}^{(i)} = g_\theta(z^{(i)})\).
   Backpropagate adversarial loss \(\frac{1}{m} \sum_{i=1}^{m} f_w(c^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(\tilde{c}^{(i)})\) and update.
Unaligned transfer for text:

Can we change an attribute (e.g. sentiment) of the text without changing the content using this autoencoder?

Example:

| Original | it has a great atmosphere, with wonderful service. |
| ARAE     | it has no taste, with a complete jerk. |
Extension: Code Space Transfer

- Extend decoder to condition on a transfer variable $\mathbf{y}$ to learn $p_\psi(\mathbf{x} \mid \mathbf{c}, \mathbf{y})$
Extension: Code Space Transfer

- Train the encoder adversarially against a classifier so that the code space is invariant to attribute $y$.
Algorithm 2 ARAE Transfer Extension

[Each loop additionally:]

(2b) **Train the code classifier** [$\min_u \mathcal{L}_{\text{class}}(\phi, u)$]
    Sample \( \{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_x \), lookup \( y^{(i)} \), and compute code-vectors \( c^{(i)} = \text{enc}_\phi(x^{(i)}) \).
    Backpropagate loss \( -\frac{1}{m} \sum_{i=1}^m \log p_u(y^{(i)} | c^{(i)}) \), update.

(3b) **Train the encoder adversarially to code classifier** [$\max_\phi \mathcal{L}_{\text{class}}(\phi, u)$]
    Sample \( \{x^{(i)}\}_{i=1}^m \sim \mathbb{P}_x \), lookup \( y^{(i)} \), and compute code-vectors \( c^{(i)} = \text{enc}_\phi(x^{(i)}) \).
    Backpropagate adversarial classifier loss \( -\frac{1}{m} \sum_{i=1}^m \log p_u(1 - y^{(i)} | c^{(i)}) \), update.
Input images are **binarized MNIST**, but normal MNIST would work as well.
Text model

AE:
\[
x \in V^n \quad \text{enc}_\phi : \text{LSTM} \quad c \in R^m \quad p_\pi : \text{LSTM} \quad x' \in V^n
\]

WGAN:
\[
z \in N(0,I) \quad \text{tanh} \quad c' \in R^m \quad \text{EM distance}
\]

\[p_\pi(x'|c) = \prod_{j=1}^n \text{softmax}(W_j h_j + b)_{x_{t,j}}\]

Not tractable

Greedy search

Same generator architecture

Text transfer model

AE:

\[ x \in V^n \]

\[ c \in R^m \]

\[ \text{enc}_\varphi : \text{LSTM} \]

\[ z \in N(0, I) \]

\[ c' \in R^m \]

\[ \text{tanh} \]

\[ g_\theta : \text{MLP} \]

\[ f_\omega : \text{MLP} \]

WGAN:

\[ \text{class} \]

\[ p_u(y | c) \]

\[ x' \in V^n \]

One decoder per class

EM distance

Same generator architecture
Experiment #1: effects of regularizing with WGAN

Checkpoint 1:
How does the norm of $c'$ behave over training?

$\textbf{c'} L2 \text{ norm} \text{ matching c L2 norm}$

[From Adversarially Regularized Autoencoders by Zhao et al, 2017]
Experiment #1: effects of regularizing with WGAN

Checkpoint 2:
How does the encoding space behave? Is it noisy?

![Graph showing variance over epochs](image)

$c'$ and $c$ sum of **dimension-wise variance** matching over time

[From Adversarially Regularized Autoencoders by Zhao et al, 2017]
Experiment #1: effects of regularizing with WGAN

Checkpoint 3:
Choose one sentence, then 100 other sentences within an edit-distance inferior to 5

Average cosine similarity in latent space. Maps similar input to nearby code.

[From Adversarially Regularized Autoencoders by Zhao et al, 2017]
Experiment #1: effects of regularizing with WGAN

Checkpoint 4:
Swap k words from an original sentence.

<table>
<thead>
<tr>
<th>k</th>
<th>AE</th>
<th>ARAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.06</td>
<td>2.19</td>
</tr>
<tr>
<td>1</td>
<td>4.51</td>
<td>4.07</td>
</tr>
<tr>
<td>2</td>
<td>6.61</td>
<td>5.39</td>
</tr>
<tr>
<td>3</td>
<td>9.14</td>
<td>6.86</td>
</tr>
<tr>
<td>4</td>
<td>9.97</td>
<td>7.47</td>
</tr>
</tbody>
</table>

[From Adversarially Regularized Autoencoders by Zhao et al, 2017]

*Left:* reconstruction error (NLL). *Right:* reconstruction examples.
Experiment #2: unaligned text transfer

Remove sentiment information from the latent space:
- At training time: adversarial training.
- At test time: pass sentences of one class, decode with the decoder from the other class

Experiment #2: unaligned text transfer

Results:

<table>
<thead>
<tr>
<th>Model</th>
<th>Transfer</th>
<th>Automatic Evaluation</th>
<th>Human Evaluation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BLEU</td>
<td>PPL</td>
</tr>
<tr>
<td>Cross-Aligned AE</td>
<td>77.1%</td>
<td>17.75</td>
<td>65.9</td>
</tr>
<tr>
<td>AE</td>
<td>59.3%</td>
<td>37.28</td>
<td>31.9</td>
</tr>
<tr>
<td>ARAE, $\lambda_a^{(1)}$</td>
<td>73.4%</td>
<td>31.15</td>
<td>29.7</td>
</tr>
<tr>
<td>ARAE, $\lambda_b^{(1)}$</td>
<td>81.8%</td>
<td>20.18</td>
<td>27.7</td>
</tr>
</tbody>
</table>

- Better transfer
- Better perplexity
- Transferred text less similar to original text

[From Adversarially Regularized Autoencoders by Zhao et al, 2017]
Experiment #3: semi-supervised classification

SNLI dataset:
- 570k human-written English sentence pairs
- 3 classes: entailment, contradiction, neutral

<table>
<thead>
<tr>
<th>Model</th>
<th>Medium</th>
<th>Small</th>
<th>Tiny</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supervised Encoder</td>
<td>65.9%</td>
<td>62.5%</td>
<td>57.9%</td>
</tr>
<tr>
<td>Semi-Supervised AE</td>
<td>68.5%</td>
<td>64.6%</td>
<td>59.9%</td>
</tr>
<tr>
<td>Semi-Supervised ARAE</td>
<td>70.9%</td>
<td>66.8%</td>
<td>62.5%</td>
</tr>
</tbody>
</table>

Medium: 22.% of labels
Small: 10.8% of labels
Tiny: 5.25% of labels

[From Adversarially Regularized Autoencoders by Zhao et al, 2017]
Playground: latent space interpolation

Idea:

GAN input: $z \in N(0,I)$

Sample 2 elements: $z_1, z_2 \sim p(z)$

For $\alpha \in [0,1]$: $z_\alpha = \alpha z_1 + (1 - \alpha)z_2$

Feed to the generator: $z_\alpha \rightarrow G \rightarrow D \rightarrow ?$

Results:

[A man is on the corner in a sport area.
A man is on corner in a road all.
A lady is on outside a racetrack.
A lady is outside on a racetrack.
A lot of people is outdoors in an urban setting.
A lot of people is outdoors in an urban setting.
A lot of people is outdoors in an urban setting.]

[From Adversarially Regularized Autoencoders by Zhao et al, 2017]
Conclusion about Adversarially Regularized AEs

Pros:

✓ Better discrete autoencoder
  - Semi-supervision
  - Text transfer

✓ Different approach to text generation

✓ Robust latent space

Cons:

❖ Sensitive to hyperparameters (GANs…)

❖ Unclear why WGAN

❖ Not so much novelty compared to Adversarial Auto Encoders (AAE)

❖ Discrete data but no discrete latent structure :/