Deep Learning

10. Generative Adversarial Networks (GANs)

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany

Syllabus

Tue. 21.4.	(1)	1. Supervised Learning (Review 1)
Tue. 28.4.	(2)	2. Neural Networks (Review 2)
Tue. 5.5.	(3)	3. Regularization for Deep Learning
Tue. 12.5.	(4)	4. Optimization for Training Deep Models
Tue. 19.5.	(5)	5. Convolutional Neural Networks
Tue. 26.5.	(6)	6. Recurrent Neural Networks
Tue. 2.6.	—	— *Pentecoste Break* —
Tue. 9.6.	(7)	7. Autoencoders
Tue. 16.6.	(8)	ctd.
Tue. 23.6.	(9)	8. Attention Layers
Tue. 30.6.	(10)	9. Graph Convolutions and Graph Attention
Tue. 7.7.	(11)	10. Generative Adversarial Networks
Tue. 14.7.	(12)	Q & A

Outline

1. Attacking Machine Learning Models

2. Adversarial Training

3. Generative Adversarial Networks

Outline

1. Attacking Machine Learning Models
2. Adversarial Training
3. Generative Adversarial Networks

What do you see?

[Szegedy et al. 2013]

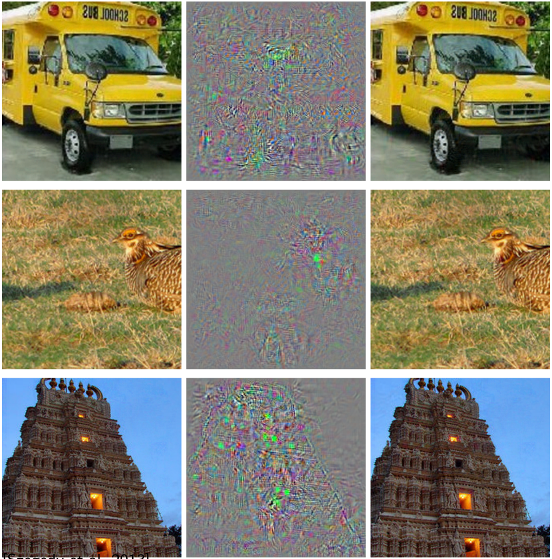
What do you see?

[Szegedy et al. 2013]

[wikipedia, art. ostrich]

AlexNet sees an ostrich.

What do you see?



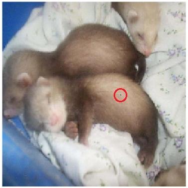
[Szegedy et al. 2013]

One Pixel Attacks

Cup(16.48%)
Soup Bowl(16.74%)

Bassinet(16.59%)
Paper Towel(16.21%)

Teapot(24.99%)
Joystick(37.39%)



Hamster(35.79%)
Nipple(42.36%)

[Su et al. 2019]

Learning Untargeted Attacks to Classifiers

Given a classifier $\hat{y} : \mathcal{X} \rightarrow \mathcal{Y}$, e.g., $\mathcal{X} := \mathbb{R}^M, \mathcal{Y} := \{0,1\}^O$ and
a pairwise loss $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$

find an attack model

$$\hat{a} : \mathcal{X} \rightarrow \mathcal{X}$$

s.t.

1. as many samples are classified **wrongly** by the classifier $\hat{y}$ after having been transformed by the attack model, i.e.,

$$\ell(\hat{a}; \mathcal{D}^{\text{test}}) := -\ell(\hat{y} \circ \hat{a}; \mathcal{D}^{\text{test}}\})$$
$$= -\frac{1}{|\mathcal{D}^{\text{test}}|} \sum_{(x,y) \in \mathcal{D}^{\text{test}}} \ell(y, \hat{y} \circ \hat{a}(x))$$

   is minimal, and

2. the attack model changes the inputs only slightly, i.e.,

$$\frac{1}{|\mathcal{D}^{\text{test}}|_{y=y^0}|} \sum_{(x,y^0) \in \mathcal{D}^{\text{test}}} ||x - \hat{a}(x)||$$

   is minimal.

Learning Targeted Attacks to Classifiers

Given a classifier $\hat{y} : \mathcal{X} \rightarrow \mathcal{Y}$, e.g., $\mathcal{X} := \mathbb{R}^M, \mathcal{Y} := \{0,1\}^O$
a pairwise loss $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$ and
a source and target label $y^0, y^1 \in \mathcal{Y}$,

find an attack model $\hat{a} : \mathcal{X} \rightarrow \mathcal{X}$
s.t.

1. as many samples from the true source class are classified as target class by the classifier $\hat{y}$ after having been transformed by the attack model, i.e.,

$$\ell(\hat{a}; \mathcal{D}^{\text{test}}) := \ell(\hat{y} \circ \hat{a}; \{(x, y^1) \mid (x, y^0) \in \mathcal{D}^{\text{test}}\}) = \frac{1}{|\mathcal{D}^{\text{test}}|_{y=y^0}|} \sum_{(x,y^0) \in \mathcal{D}^{\text{test}}} \ell(y^1, \hat{y} \circ \hat{a}(x))$$

   is minimal, and

2. the attack model changes the inputs only slightly, i.e.,

$$\frac{1}{|\mathcal{D}^{\text{test}}|_{y=y^0}|} \sum_{(x,y^0) \in \mathcal{D}^{\text{test}}} ||x - \hat{a}(x)||$$

   is minimal.

Additive Attacks

- additive attack models:

$$\hat{a}(x) := x + \hat{\epsilon}(x), \quad \hat{\epsilon} : \mathcal{X} \to \mathcal{X}$$
$$\ell(y^1, \hat{y} \circ a(x)) = \ell(y^1, \hat{y}(x + \hat{\epsilon}(x)))$$
$$||x - \hat{a}(x)|| = ||\hat{\epsilon}(x)||$$

- use maximum norm $||\hat{\epsilon}(x)||_\infty$
- instead of minimizing $||\hat{\epsilon}(x)||_\infty$, enforce

$$||\hat{\epsilon}(x)||_\infty < \epsilon_{\text{max}}, \quad \forall x \in X, \quad \text{for } \epsilon \in \mathbb{R}^+$$

- **being attackable**

$$\forall (x, y^0) \in \mathcal{D}\, \exists \hat{\epsilon}(x) : ||\hat{\epsilon}(x)|| < \epsilon_{\text{max}}, \quad \hat{y}(x + \hat{\epsilon}(x)) = y^1$$

is different from **being unstable**

$$\forall (x, y) \in \mathcal{D} : \; p(\hat{y}(x + \epsilon) \neq \hat{y}(x) \mid \epsilon \sim \mathcal{X}, ||\epsilon|| < \epsilon_{\text{max}})$$

Fast Gradient Sign Attack

- very simple untargeted attack [Goodfellow et al., 2014]
- idea: for a linear model

$$\hat{y}(x+\hat{\epsilon}) = w^T(x+\hat{\epsilon}) = w^T x + w^T \hat{\epsilon}$$

grows maximally (under constaint $\hat{\epsilon} \leq \epsilon_{\max}$) for $\hat{\epsilon} := \epsilon_{\max} \operatorname{sgn}(w)$

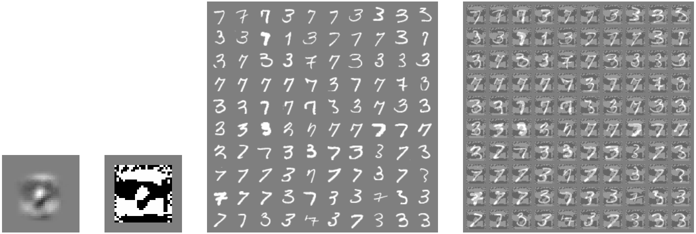
$$= \hat{y}(x) + \epsilon_{\max} ||w||_1$$

- for a non-linear model:

$$\hat{\epsilon}(x,y) := \epsilon_{\max} \operatorname{sgn}(\nabla_x(\ell(y,\hat{y}(x))))$$

- can be computed by backpropagation
- simple heuristics
- requires knowledge of the attacked model $\hat{y}$ (whitebox)

Fast Gradient Sign Attack / Examples



(a) (b) (c) (d)

[Goodfellow et al. 2014]

a) weights of a logistic regression model
b) their sign (= gradient sign for any x), i.e., the best attack
c) original examples for 3s and 7s (1.6% error)
d) attacked examples (99% error)

Outline

1. Attacking Machine Learning Models

2. Adversarial Training

3. Generative Adversarial Networks

Adversarial Training

- can we make a model more robust against attacks?
- idea:
  1. augment training data by **adversarial examples** $\hat{a}(x)$ with correct class y:

$$\text{aug}(\mathcal{D}^{\text{train}}) := \{(\hat{a}(x), y) \mid (x, y) \in \mathcal{D}^{\text{train}}\}$$

    - as aug depends on the attack model $\hat{a}$, which in turn depends on $\hat{y}$, the augmented dataset will shift during training of $\hat{y}$.
    - think about it as a generator / distribution.

  2. train on both parts of the data:

$$\ell(\hat{y}; \mathcal{D}^{\text{train}}, \text{aug}) := \ell(\hat{y}; \mathcal{D}^{\text{train}}) + \alpha\, \ell(\hat{y}; \text{aug}(\mathcal{D}^{\text{train}}))$$

    - α is a hyperparameter.
    - Goodfellow et al. 2014 uses $\alpha = 1$.

Adversarial Training / Results

- MNIST dataset

model $\hat{y}$	trained on	error [%] on own adv. ex.	others adv. ex.	orig. ex.
small maxout net	$\mathcal{D}^{\text{train}}$	89.4	40.9	0.94
small maxout net	$\mathcal{D}^{\text{train}}, \text{aug}(\mathcal{D}^{\text{train}})$	17.9	19.6	0.84
large maxout net	$\mathcal{D}^{\text{train}}$			1.14
large maxout net	$\mathcal{D}^{\text{train}}, \text{aug}(\mathcal{D}^{\text{train}})$			0.782
small maxout net	$\mathcal{D}^{\text{train}}, \pm\epsilon$	86.2		
small maxout net	$\mathcal{D}^{\text{train}}, \text{unif}(-\epsilon, +\epsilon)$	90.4		

- adversarial training dampens a models attackability considerably.
  - also for adversarial examples transferred from other models.
- adversarial training can have a regularizing effect !

Note: adv.=adversarial, ex.= examples.

Learning to Augment Data

Given a training dataset $\mathcal{D}^{\text{train}} \in (\mathcal{X} \times \mathcal{Y})^*$,
a pairwise loss $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$,
a learning algorithm $L : (\mathcal{X} \times \mathcal{Y})^* \to \mathcal{Y}^{\mathcal{X}}$

find a data augmentation model

$$\hat{a} : (\mathcal{X} \times \mathcal{Y})^* \to (\mathcal{X} \times \mathcal{Y})^*$$

s.t. the model learned on the augmented data has a minimal loss:

$$\begin{aligned}\ell(\hat{a}; \mathcal{D}^{\text{test}}) &:= \ell((L \circ \hat{a})(\mathcal{D}^{\text{train}}); \mathcal{D}^{\text{test}}) \\ &= \frac{1}{|\mathcal{D}^{\text{test}}|} \sum_{(x,y) \in \mathcal{D}^{\text{test}}} \ell(y, \hat{y}(x)), \quad \hat{y} := L(\hat{a}(\mathcal{D}^{\text{train}}))\end{aligned}$$

Outline

1. Attacking Machine Learning Models

2. Adversarial Training

3. Generative Adversarial Networks

Learning Distributions I: Density Estimation

- **density estimation**:
  given a sample $\mathcal{D}^{\text{train}} \subset \mathcal{X}$ of instances sampled from an unknown distribution $p : \mathcal{X} \to \mathbb{R}_0^+$,
  learn the density function

$$\hat{p} : \mathcal{X} \to \mathbb{R}_0^+$$

  i.e., a function that assigns each instance $x \in \mathcal{X}$ a likelihood $\hat{p}(x)$,
  s.t. the integral of $\hat{p}$ over any measurable subset $X \subset \mathcal{X}$
  yields the average number of instances in X in fresh samples $\mathcal{D}^{\text{test}} \sim p$

$$\int_X \hat{p}(x)dx \overset{!}{\approx} \int_X p(x)dx = \mathbb{E}_{x\sim p}(x \in X) \quad \forall X \subseteq \mathcal{X} \text{ measurable}$$

  - this construction does not allow to compute new samples from $\hat{p}$ directly.

Learning Distributions II: Generative Models

- **generative model**:
  given a sample $\mathcal{D}^{\text{train}} \subset \mathcal{X}$ of instances sampled from an unknown distribution $p : \mathcal{X} \rightarrow \mathbb{R}_0^+$,
  learn a **generative model**

$$q : Z \rightarrow \mathbb{R}_0^+$$
$$\hat{x} : Z \rightarrow \mathcal{X}$$

  - where q is a distribution on Z that is easy to sample from, often just the multivariate standard normal $q := \mathcal{N}_K(0, \text{diag}(1, \ldots, 1))$
  - s.t. the average number of instances in fresh samples $\mathcal{X}^{\text{test}} \sim p$ that fall within any measurable subset $X \subset \mathcal{X}$, are just the integral of p over X:

$$\mathbb{E}_{z \sim q}(\hat{x}(z) \in X) \stackrel{!}{\approx} \int_X p(x)dx = \mathbb{E}_{x \sim p}(x \in X) \quad \forall X \subseteq \mathcal{X} \text{ measurable}$$

  - this construction allows to generate new samples x via

$$z \sim q, \quad x := \hat{x}(z)$$

Generative Models

Learn to generate data that looks as close as possible to a real dataset:

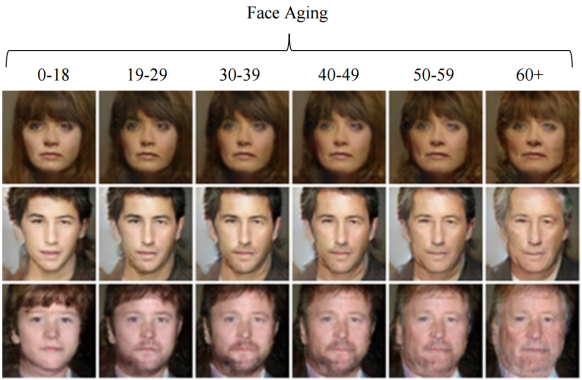


Figure 1: Conditional Generation, Source: Antipov 2017

Generation as a Maximum Likelihood Task

- Maximize the likelihood of observing the data using parameters θ:

$$
\begin{aligned}
\theta^* &= \arg\max_\theta \prod_{n=1}^{N} \hat{p}(x^{(n)};\theta) \\
&= \arg\max_\theta \log \prod_{n=1}^{N} \hat{p}(x^{(n)};\theta) \\
&= \arg\max_\theta \sum_{n=1}^{N} \log \hat{p}(x^{(n)};\theta) \\
&\approx \arg\max_\theta \mathbb{E}_{x\sim p}(\log \hat{p}(x;\theta)) \\
&= \arg\min_\theta \mathbb{E}_{x\sim p}(-\log \hat{p}(x;\theta)) \\
&= \arg\min_\theta \int p(x) \log \frac{p(x)}{\hat{p}(x;\theta)} dx =: D_{KL}(p \,||\, \hat{p}(.;\theta))
\end{aligned}
$$

- equivalent to minimize the **Kullback Leibler divergence** between the true distribution p and the estimated distribution $\hat{p}$.

Generative Adversarial Networks (GAN)

- Two agents play a minimax game:
  - **Generator**: Generate synthetic data aiming to make them as similar as possible to real data
  - **Discriminator**: Distinguish if an input sample comes from the real data distribution

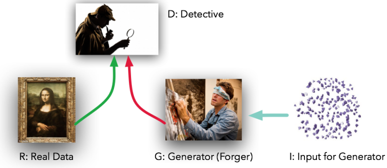


Figure 2: GAN, Courtesy of Dev Nag

- *Generator* **MIN**imizes the following:
  - *Discriminator* **MAX**imizes the accuracy of counterfeit detection

GAN - Problem

- Unknown distribution p over data instances $x \in \mathbb{R}^M$
- **Generator**:
  - generate new instances, implicitly defines distribution $\hat{p} : \mathbb{R}^M \rightarrow \mathbb{R}_0^+$
  - $\hat{x}(., \theta_g) : \mathbb{R}^K \rightarrow \mathbb{R}^M$ is a neural network.
  - $z \sim q : \mathbb{R}^K \rightarrow \mathbb{R}_0^+$ sometimes called noise or prior.
- **Discriminator**:
  - $d(x, \theta_d) : \mathbb{R}^M \rightarrow [0, 1]$ is a neural network
  - $d(x)$ is the probability that x comes from real data rather than being generated by $\hat{x}$.
- GANs aim to learn θ_g and θ_d optimizing a joint objective:

$$\min_{\theta_g} \max_{\theta_d} \; \mathbb{E}_{x \sim p}(\log d(x; \theta_d)) + \mathbb{E}_{z \sim q}(\log(1 - d(\hat{x}(z; \theta_g); \theta_d)))$$

GAN - Optimization

learn-gan$(\mathcal{D}^{\text{train}}, B, I, I^{\text{discrim}})$:

  initialize θ_d, θ_g

  for I iterations :

    for I^{discrim} iterations:

      sample B noise samples: $z_1, \ldots, z_B \sim q$

      sample B real samples: $x_1, \ldots, x_B \sim \mathcal{D}^{\text{train}}$

      update discriminator parameters θ_d using gradient ascent:

$$\nabla_{\theta_d} \frac{1}{B} \sum_{b=1}^{B} \log d(x_b; \theta_d) + \log(1 - d(\hat{x}(z_b; \theta_g); \theta_d))$$

    sample B noise samples: $z_1, \ldots, z_B \sim q$

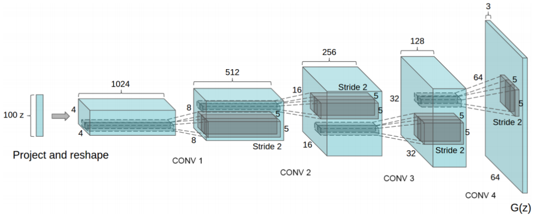
    update generator parameters θ_g using gradient descent:

$$\nabla_{\theta_g} \frac{1}{B} \sum_{b=1}^{B} \log(1 - d(\hat{x}(z_b; \theta_g); \theta_d))$$

  return θ_d, θ_g

Deep Convolutional Generative Adversarial Networks

- Replace pooling with strided convolutions (discriminator) and fractional-strided convolutions (generator)
- Use batchnorm in both generator and discriminator
- Remove fully connected hidden layers
- Use ReLU in generator for all layers, except output (tanh)
- Use LeakyReLU in discriminator for all layers



DCGAN / Example

Figure 4: DCGAN Generated Images discriminated against the LSUN dataset, Source: Radford et al., ICLR 2016

Summary

- Machine Learning Models can be **attacked**, i.e., is possible for any instance,
  - to modify it only slightly (imperceptible),
  - but s.t. the model predicts an arbitrary class.

- The **Fast Gradient Sign Attack** is a simple such attack that moves instances in the direction of the elementwise sign of their gradients.

- **Adversarial training**, i.e., include adversarial samples and their true class into the training set, can help to mitigate the impact of attacks somewhat.

- **Generative Adversarial Networks** aim to learn to generate new instances, by optimizing a joint loss for
  - a **generator model**, that creates/reconstructs instances from a latent representation (that is easy to sample), and
  - a **discriminator model** that aims to distinguish true from generated samples.

Further Readings

- Zhang et al. 2020, ch. 17 covers some basic principles.
- Goodfellow et al. 2016, ch. 7.13 briefly covers adversarial training.
- a survey: Akhtar and Mian [2018]
- a library: cleverhans.
  https://github.com/tensorflow/cleverhans

Acknowledgement: An earlier version of the slides for this lecture have been written by my former postdoc Dr Josif Grabocka.

References

Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6: 14410–14430, 2018.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The Mit Press, Cambridge, Massachusetts, November 2016. ISBN 978-0-262-03561-3.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Jiawei Su, Danilo Vasconcellos Vargas, and Sakurai Kouichi. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, October 2019. ISSN 1089-778X, 1089-778X, 1941-0026. doi: 10.1109/TEVC.2019.2890858.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander Smola. *Dive into Deep Learning*. https://d2l.ai/, 2020.