SCHOOL OF DATA ANALYSIS

LAMBDA

Yandex
Data Factory

Generative Adversarial Networks

A metamorphosis ~~magic~~ mathematics

Andrey Ustyuzhanin, Maxim Borisyak, Mikhail Usvyatsov, Alexander Panin

Yandex School of Data Analysis
Yandex Data Factory

Laboratory of Methods for Big Data Analysys
National Research University Higher School of Economics

Generative

Generative

> Given a dataset D, generate samples similar to these from D.

Mathematically, construct a random variable X' (and corresponding sampling procedure) that has distribution close to these of X:

$$P_{X'} \approx P_X$$

'Statistical' approach

- introduce a parametrized probability distribution family $P_\theta(x)$;
- fit the distribution:

$$\begin{aligned}\mathcal{L}_\theta(D) &= \prod_{x \in D} P_\theta(x)\\ \theta^* &= \arg\max_\theta \mathcal{L}_\theta(D)\end{aligned}$$

- sample from P_{θ^*};
- profit.

Deep Learning approach, first attempt

- introduce a parametrized probability distribution family $P_\theta(x)$:
    - introduce latent variables V and a function (network) to produce x from V (classification in reverse);
- fit the distribution:
    - train the network;
- sample from P_{θ^*};
- profit.

Deep Learning approach, first attempt

- it is easy to define a model for 'scores' (unnormalized probabilities):

$$\begin{aligned} P(x) &= \frac{1}{Z} s(x) \\ Z &= \mathrm{const} \end{aligned}$$

- normalization might be a problem:

$$Z = \int s(x) dx$$

or

$$Z = \sum_x s(x)$$

Deep Learning approach, first attempt

- in popular models normalization constant changes with change in parameters;
- tractably compute updates with regard to normalization coefficient might be hard;
- e.g. RBM (one of such models) has to run long Monte-Carlo Markov sampling chains to make an updates by one (!) sample.

> This approach is possible but might be complicated in practice.

Adversarial

Adversarial

Let's rewind to the original problem.

> Find a sampling procedure for X':
>
> $$P_{X'} \approx P_X$$

Let's reformulate problem a little bit:

$$\rho(P_{X'}, P_X) \to_{P_{X'}} \min$$

Adversarial

- let's introduce some latent variables with fixed distribution e.g.:

$$V \sim U^n[-1, 1]$$

- and a parametrized generation procedure:

$$X' = g_\theta(V)$$

Adversarial

Some reformulation:

$$\rho(P_{X'}, P_X) \quad \to_{P_{X'}} \quad \text{min}$$

$$\rho(P_{g_\theta(V)}, P_X) \quad \to_{g_\theta(V)} \quad \text{min}$$

$$\rho(P_{g_\theta(V)}, P_X) \quad \to_{\theta} \quad \text{min}$$

Adversarial

> What can be used as a distance measure ρ between two distributions?
> (One of which is defined as a dataset.)

Adversarial

> A classifier would be a good statistical similarity measure.
> - seconds before invention of GAN.

$$\rho(P_{X'}, P_X) \to \text{min} \iff \text{trained classifier loss} \to \text{max}\,.$$

Adversarial

- let's define two network:
  - $d_\zeta(x)$ - classifier to measure distance, **discriminator**;
  - $g_\theta(x)$ - network to transform latent variables V to X', **generator**;
- loss function of discriminator (e.g. cross-entropy):

$$\begin{aligned}
L(X, X') &= \frac{1}{2}\mathbb{E}_{x\sim X} l(d_\zeta(x), 1) + \frac{1}{2}\mathbb{E}_{x'\sim X'} l(d_\zeta(x'), 0) \\
&= -\frac{1}{2}\left(\mathbb{E}_{x\sim X} \log d_\zeta(x) + \mathbb{E}_{x'\sim X'} \log(1 - d_\zeta(x'))\right) \\
&= -\frac{1}{2}\left(\mathbb{E}_{x\sim X} \log d_\zeta(x) + \mathbb{E}_{v\sim V} \log(1 - d_\zeta(g_\theta(v)))\right)
\end{aligned}$$

Adversarial

Distributions X and V are fixed:

$$L(X, X') = -\frac{1}{2}\left(\mathbb{E}_{x \sim X} \log d_\zeta(x) + \mathbb{E}_{v \sim V} \log(1 - d_\zeta(g_\theta(v)))\right)$$

$$= L(\theta, \zeta)$$

Adversarial

Back to the problem:

$$\rho(P_{X'}, P_X) \to \text{min} \iff \text{trained classifier loss} \to \text{max}$$

Trained classifier loss:

$$\text{trained classifier loss} = L^*(\theta) = \min_{\zeta} L(\zeta, \theta)$$

Adversarial

Trained classifier loss:

$$\text{trained classifier loss} \to \max$$

$$\min_{\zeta} L(\zeta, \theta) \to_{\theta} \max$$

$$\theta^* = \arg\max_{\theta} \left[\min_{\zeta} L(\zeta, \theta) \right]$$

Network

Network

$$\theta^* = \arg\max_\theta \left[\min_\zeta L(\zeta, \theta) \right]$$

› let's define the optimal discriminator:

$$\begin{aligned} d^*_\theta &= d_{\zeta^*(\theta)} \\ \zeta^*(\theta) &= \arg\min_\zeta L(\zeta, \theta) \end{aligned}$$

Network

$$\theta^* = \arg\max_\theta \left[\min_\zeta L(\zeta, \theta)\right]$$

- training generator with SGD:

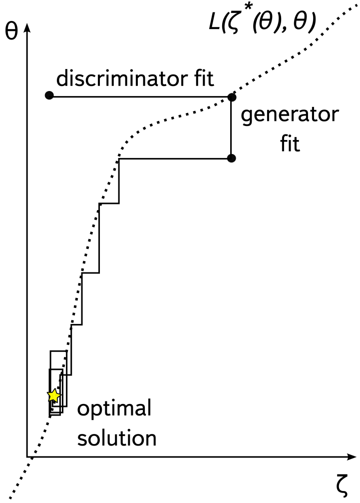
$$\Delta\theta \sim \nabla L(\zeta^*(\theta), \theta)$$

- for small changes $\Delta\theta$ in θ:

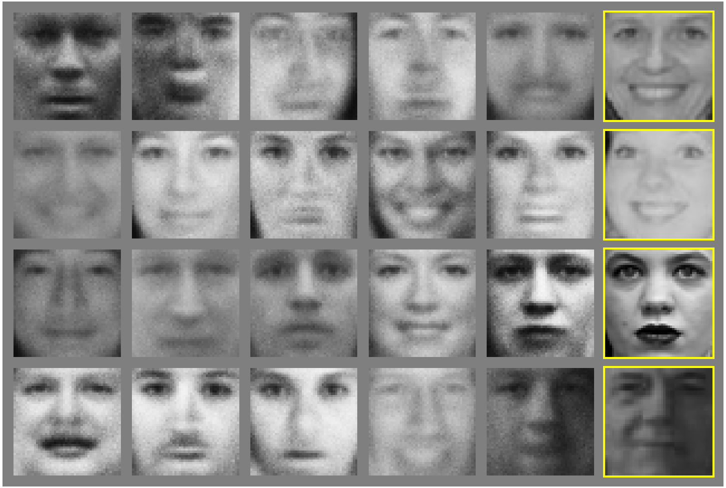
$$\nabla L(\zeta^*(\theta), \theta) \approx \nabla L(\zeta^*(\theta), \theta + \Delta\theta)$$

Training strategy

- train discriminator to nearly optimal under constant generator;
- make a small changes in generator under constant discriminator;
- process may cycle;
- repeat until bored.

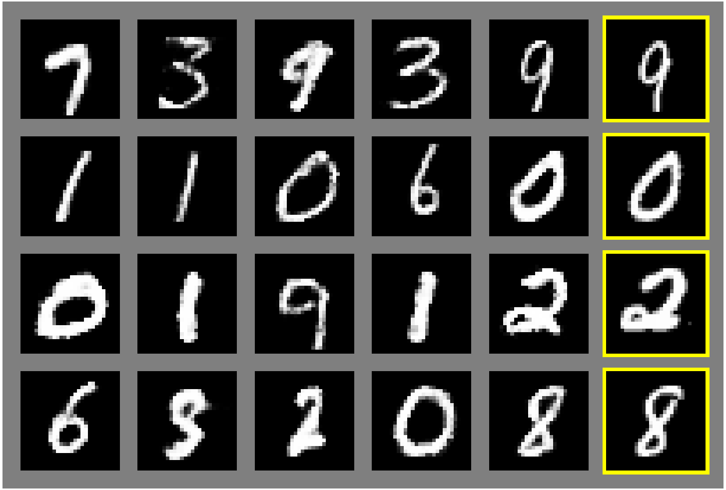


Examples



Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

Examples



Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

Discussion

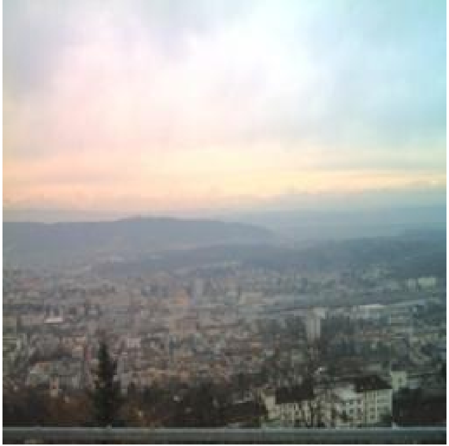
- model for distribution is implicitly set by choice of discriminator:
  - easy to formulate what kind of similarity one wants from generator;
  - without explicitly formulating distribution family;
  - or constructing specific generator;

Discussion

- easily allows modifications:
    - conditional GAN:
    - mixing GAN objective with others;
    - training a domain invariant networks

Turn day into night

input

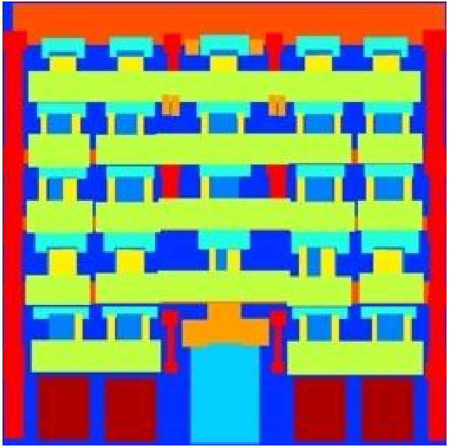


output

Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." arXiv preprint arXiv:1611.07004 (2016).

Auto-architect

input

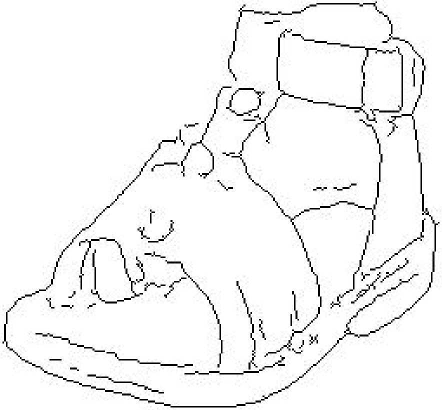


output

Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." arXiv preprint arXiv:1611.07004 (2016).

Make a shoe of your dreams!

input



output

Isola, Phillip, et al. "Image-to-image translation with conditional adversarial networks." arXiv preprint arXiv:1611.07004 (2016).

Summary

Summary

- Generative Adversarial Networks:
  - generator training is driven by classifier:
    - two-step optimization;
  - all assumptions are set implicitly by classifier:
    - usually easier than explicit generator construction;
  - allow a wide range of modifications.