SCHOOL OF DATA ANALYSIS

LAMBDA

Generative Adversarial Networks

part 2

Maxim Borisyak, Alexander Panin

Yandex School of Data Analysis
National Research University Higher School of Economics

Laboratory of Methods for Big Data Analysys

Classical GAN

Quick recap

$$\mathcal{L} = \frac{1}{2}\left(\mathcal{L}_{\mathrm{real}} + \mathcal{L}_{\mathrm{pseudo}}\right) =$$

$$- \frac{1}{2}\left(\mathop{\mathbb{E}}_{X \sim \mathrm{Real}} \log(D(x)) + \mathop{\mathbb{E}}_{Z \sim \dots} \log(1 - D(G(Z_i)))\right)$$

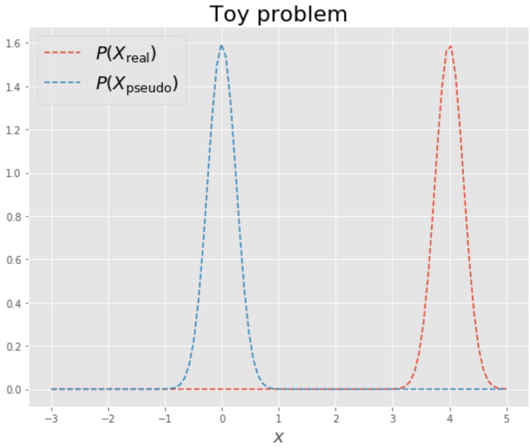
discriminator: $\mathcal{L} \to \min$

generator: $\mathcal{L}_{\mathrm{pseudo}} \to \max$

Learning from a strict teacher

Consider GAN:

- powerful discriminator;
- initially disjoint supports of generated and real samples:
  - e.g. images of cats lie on a low-dimensional subpace;
  - randomly initialized generator is likely to miss the subspace.



Learning from a strict teacher

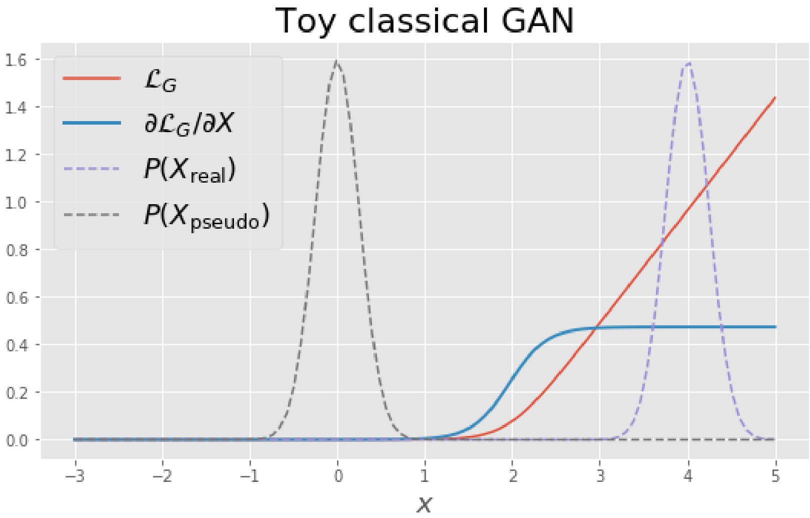
Generator gradients tend to vanish on early stages.

$$\frac{\partial \mathcal{L}_{\mathrm{pseudo}}}{\partial G} = \frac{\partial \mathcal{L}_{\mathrm{pseudo}}}{\partial D} \frac{\partial D}{\partial G} = \frac{1}{2} \mathop{\mathbb{E}}_{Z \sim \dots} \frac{1}{1 - D(G(Z))} \frac{\partial D(G)}{\partial G};$$

$$D(G(Z)) \approx 0;$$

$$\frac{\partial D(G)}{\partial G} \approx 0.$$

Learning from a strict teacher



Fight for the gradients

Use gradient-free optimization methods:

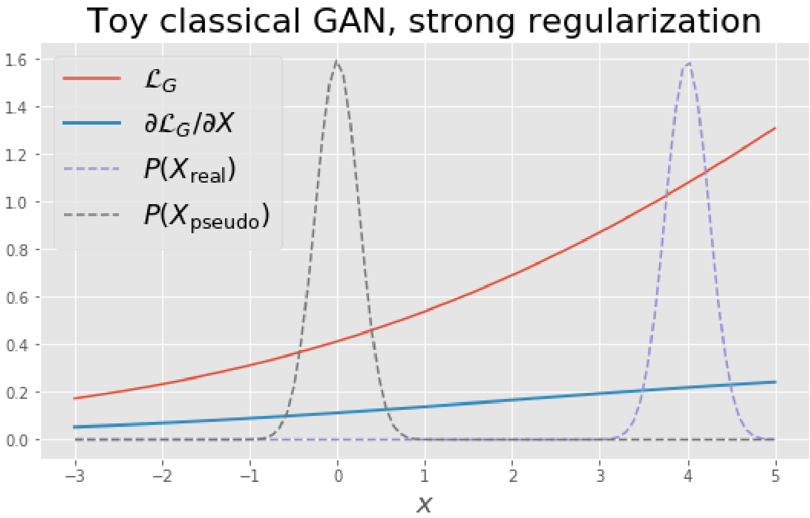
- degrade for high-dimensional problems (i.e. almost all Deep Learning problems);
- computationally costly;
- usually does not go along with stochasticity;
- discriminator has a chance to get stuck.

Fight for the gradients

Introduce strong regularization to make discriminator less powerful:

- strong regularization hurts in late stages;
- adjusting regularization can be tricky.

Strong regularization



Fight for the gradients

Early stopping:

- hard to control;
- similar to strong regularization.

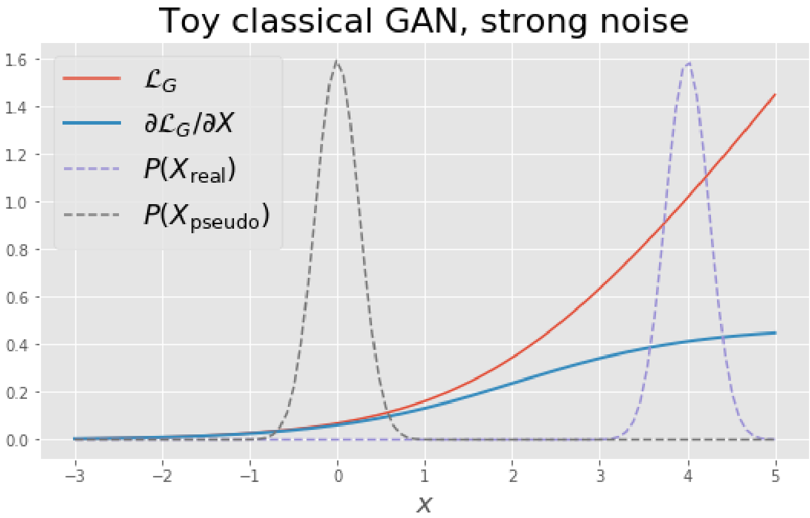
```
for epoch in ...:
  while loss > C1:
    train_discriminator()

  while loss < C2:
    train_generator()
```

Fight for the gradients

Introduce noise into discriminator inputs:

- similar to regularization;
- reduces quality of solution;
- adjusting noise level can be tricky.

Noisy inputs



Fight for the gradients

Make a ensemble of discriminators $\{D^i\}_{i=1}^n$ with increasing power:

$$\mathcal{L} = \frac{1}{2n} \sum_{i=1}^{n} \left(\mathcal{L}^i_{\mathrm{real}} + \mathcal{L}^i_{\mathrm{pseudo}}\right) =$$

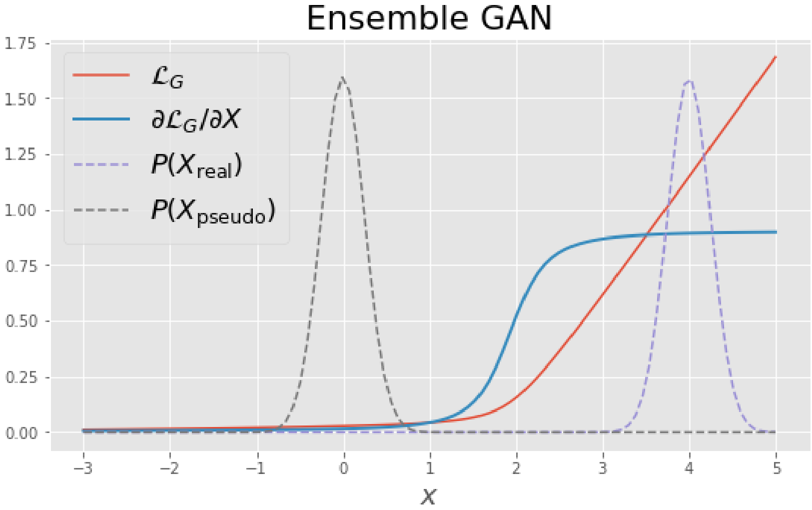
$$- \frac{1}{2n} \sum_{i=1}^{n} \left[\mathop{\mathbb{E}}_{X \sim \mathrm{Real}} \log(D^i(x)) + \mathop{\mathbb{E}}_{Z \sim \dots} \log(1 - D^i(G(Z_i))) \right]$$

Fight for the gradients

Make a ensemble of discriminators $\{D^i\}_{i=1}^n$ with increasing power:

- always provide gradients;
- convergence to the same solution as only with the strongest discriminator.

Discriminator ensemble



Bonus: generator quality assessment

Option 1: output of generator is used for another problem that has well-defined quality metric. Option 2: no such metric is availible:

- likelihood is hard to compute and is deeply flawed for this case;
- any statistical distance requires probabilistic model;
  - thus, no reason for using GAN if such model is available for fitting.

Bonus: generator quality assessment

In GAN discriminator serves as proxy for probabistic model:

- losses of a fixed discriminator can serve as a quality metric.

Consider an ensemble of discriminators $\{D_i\}_i$:

- strictly increasing discriminative power;
- for each $i > j$ D_i strictly dominates D_j everywhere;

Then:

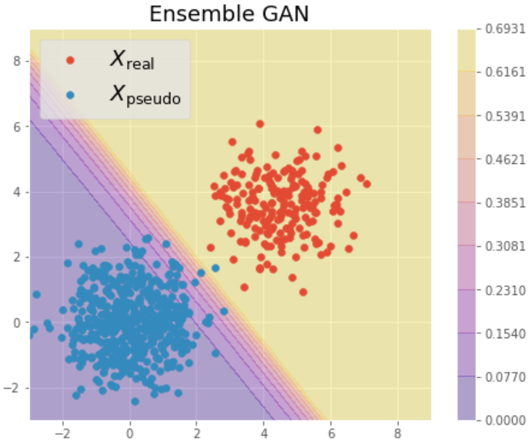
- for each $i > j$ $\mathcal{L}_i < \mathcal{L}_j$.

Ensemble profile

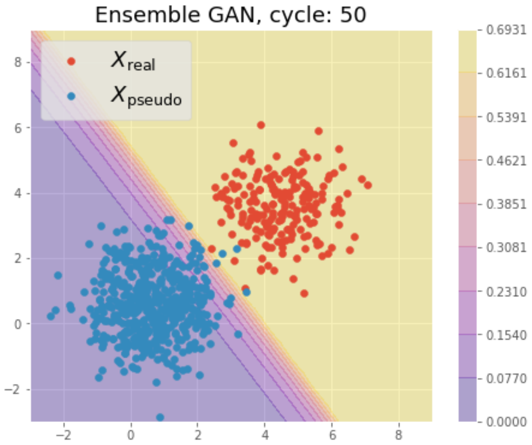
$\{\mathcal{L}_i\}_i$ can be used for a fine quality assesment.

Toy example

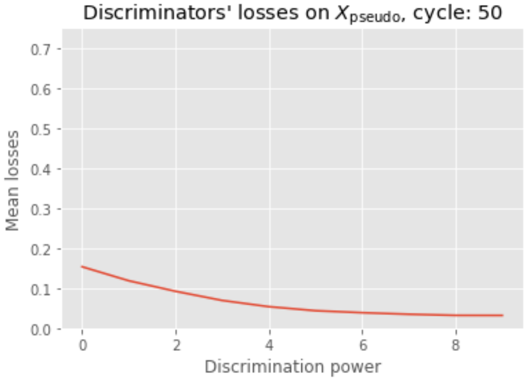
- $X_{\text{real}} \sim \mathcal{N}(\mu, 1)$;
- $X_{\text{pseudo}} = Z + m$;
- $Z \sim \mathcal{N}(0, 1)$;
- disciriminators:
  - 10 logistic regressions with Gaussian noise;
  - noise σ decreases from 2 to 0.



Toy example

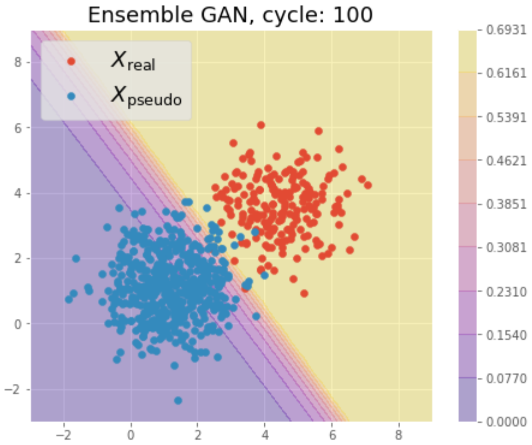


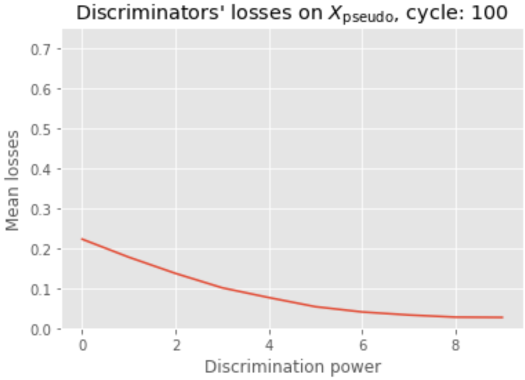
Ensemble GAN, cycle: 50
X_{real}
X_{pseudo}
8
6
4
2
0
−2
−2
0
2
4
6
8
0.6931
0.6161
0.5391
0.4621
0.3851
0.3081
0.2310
0.1540
0.0770
0.0000



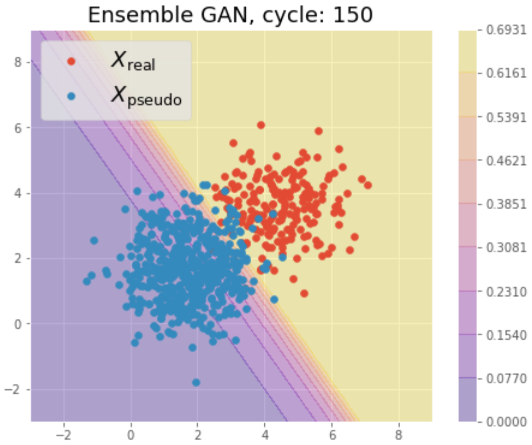
Discriminators' losses on X_{pseudo}, cycle: 50
Mean losses
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8
Discrimination power

Toy example

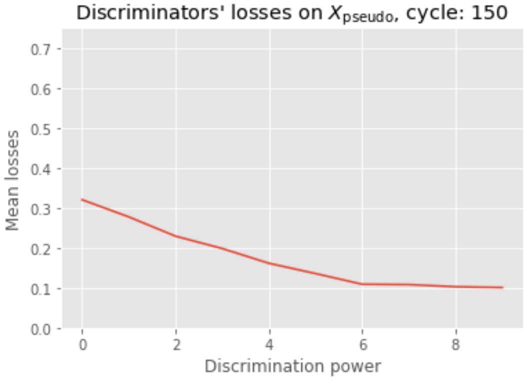

Ensemble GAN, cycle: 100
X_{real}
X_{pseudo}
8
6
4
2
0
−2
−2
0
2
4
6
8
0.6931
0.6161
0.5391
0.4621
0.3851
0.3081
0.2310
0.1540
0.0770
0.0000


Discriminators' losses on X_{pseudo}, cycle: 100
Mean losses
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8
Discrimination power

Toy example

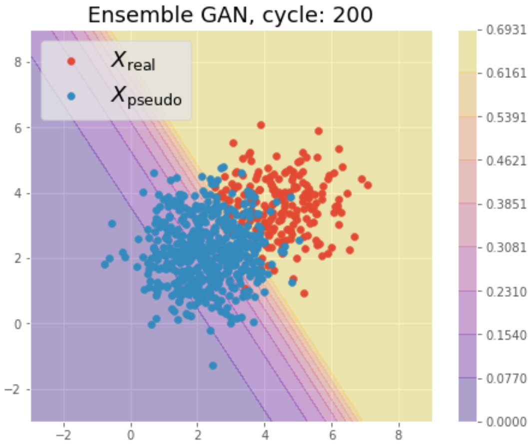


Ensemble GAN, cycle: 150
X_{real}
X_{pseudo}
8
6
4
2
0
−2
−2
0
2
4
6
8
0.6931
0.6161
0.5391
0.4621
0.3851
0.3081
0.2310
0.1540
0.0770
0.0000

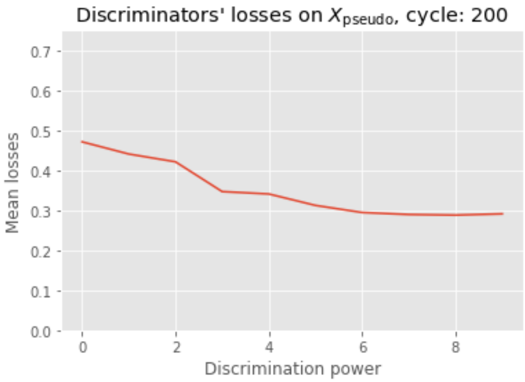


Discriminators' losses on X_{pseudo}, cycle: 150
Mean losses
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8
Discrimination power

Toy example

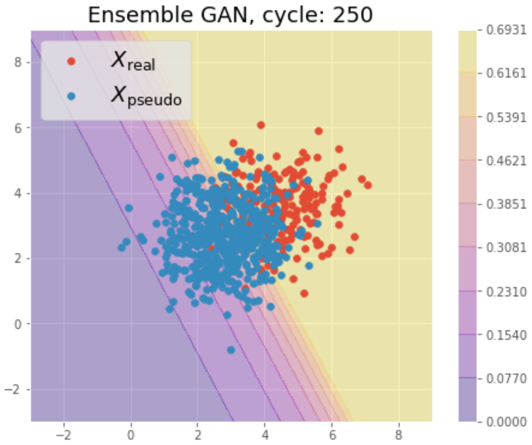


Ensemble GAN, cycle: 200
X_{real}
X_{pseudo}
8
6
4
2
0
−2
−2
0
2
4
6
8
0.6931
0.6161
0.5391
0.4621
0.3851
0.3081
0.2310
0.1540
0.0770
0.0000

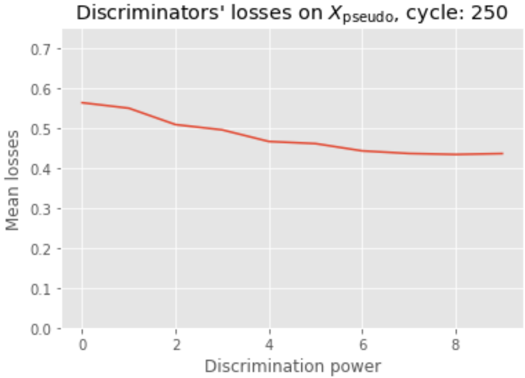


Discriminators' losses on X_{pseudo}, cycle: 200
Mean losses
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8
Discrimination power

Toy example

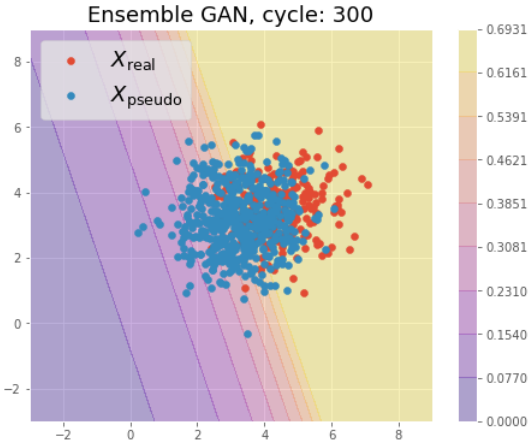


Ensemble GAN, cycle: 250
X_{real}
X_{pseudo}
8
6
4
2
0
−2
−2
0
2
4
6
8
0.6931
0.6161
0.5391
0.4621
0.3851
0.3081
0.2310
0.1540
0.0770
0.0000

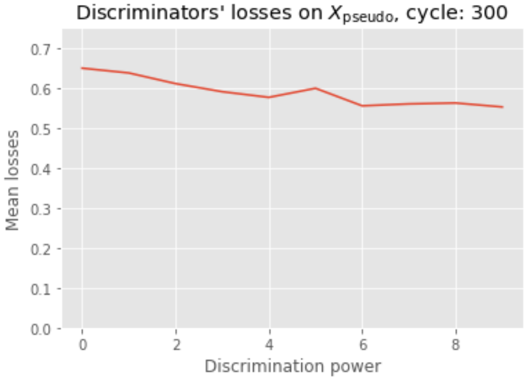


Discriminators' losses on X_{pseudo}, cycle: 250
Mean losses
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8
Discrimination power

Toy example

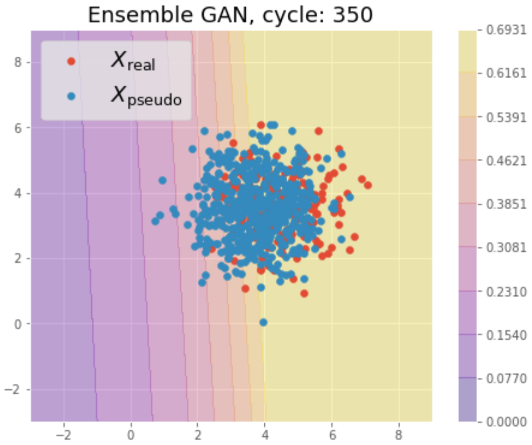


Ensemble GAN, cycle: 300
X_{real}
X_{pseudo}
8
6
4
2
0
−2
−2
0
2
4
6
8
0.6931
0.6161
0.5391
0.4621
0.3851
0.3081
0.2310
0.1540
0.0770
0.0000

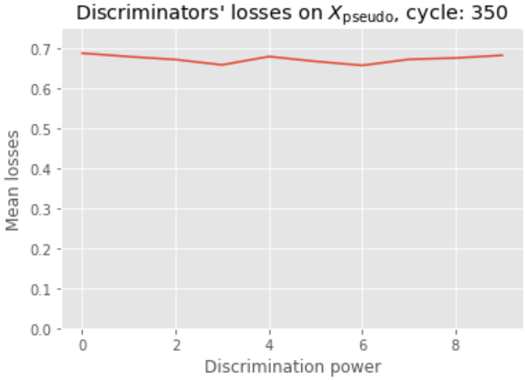


Discriminators' losses on X_{pseudo}, cycle: 300
Mean losses
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8
Discrimination power

Toy example

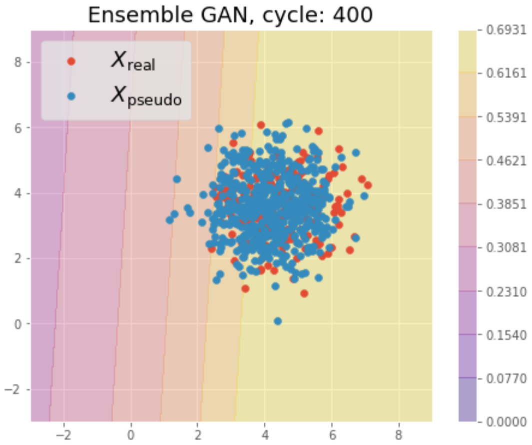


Ensemble GAN, cycle: 350
X_{real}
X_{pseudo}
8
6
4
2
0
−2
−2
0
2
4
6
8
0.6931
0.6161
0.5391
0.4621
0.3851
0.3081
0.2310
0.1540
0.0770
0.0000

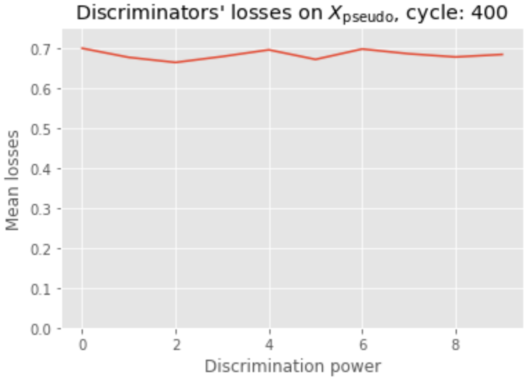


Discriminators' losses on X_{pseudo}, cycle: 350
Mean losses
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8
Discrimination power

Toy example



Ensemble GAN, cycle: 400
X_{real}
X_{pseudo}
8
6
4
2
0
−2
−2
0
2
4
6
8
0.6931
0.6161
0.5391
0.4621
0.3851
0.3081
0.2310
0.1540
0.0770
0.0000



Discriminators' losses on X_{pseudo}, cycle: 400
Mean losses
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
2
4
6
8
Discrimination power

Energy-based GAN

Energy Models

Energy models instead of predicting probabilities, distribute 'energy' $E(x)$:

- usually energy can be transformed to probability, as e.g.:

$$P(x) = \frac{\exp[-E(x)]}{Z};$$
$$Z = \int_{x'} \exp\left[-E(x')\right] dx'.$$

- one of the simpliest energy models is AutoEncoder:

$$E(x) = \|\mathrm{decode}(\mathrm{encode}(x)) - x\|_2^2$$

Energy-based GAN

- discrimination with energy models can be done by maximizing energy gap:

$$\mathcal{L} = \mathop{\mathbb{E}}_{X \sim \mathcal{C}_1} E(X) + \mathop{\mathbb{E}}_{X \sim \mathcal{C}_2} \left[m - E(X)\right]_+ \to \min$$

where:

- $[f]_+ = \max(0, f)$;
- $\mathcal{C}_1, \mathcal{C}_2$ - classes;
- m - margin, hyper-parameter.

Denormalization

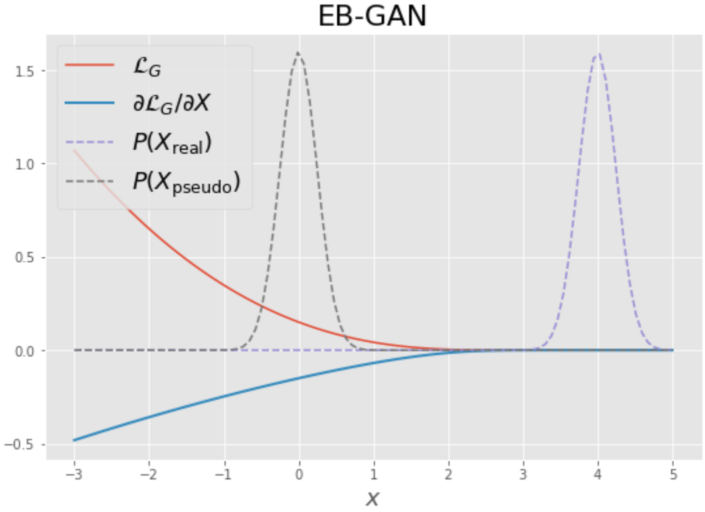
Energy-based GAN:

discriminator: $$\mathop{\mathbb{E}}_{X \sim \mathrm{real}} E(X) + \mathop{\mathbb{E}}_{Z \sim \dots} \left[m - E(D(Z))\right]_+ \to \min;$$

generator: $$\mathop{\mathbb{E}}_{Z \sim \dots} E(D(Z)) \to \min.$$

- absence of normalization leads to non-vanishing gradients.

Energy-based GAN



Discussion

- discriminator in EB-GAN are trained as long as it is possible;
- EB-GAN doesn't require any tricks to work;
- in most cases, superior to classical GAN;
- in most cases, computationally faster:
  - $2\times$ slower due to AutoEncoder architecture;
  - compensated by faster convergence.

Wasserstein GAN

Generalizing GAN approach

GAN discriminator is a proxy for a statistical distance:

- classical GAN, Kullback-Leibler is minimized (thus vanishing gradients);
- EB-GAN minimizes total variation distance.

Wasserstein GAN minimizes Earth-mover distance:

$$W(P_X, P_Y) = \inf_{\gamma \in \Pi(P_X, P_Y)} \mathop{\mathbb{E}}_{X, Y \sim \gamma} \|X - Y\|$$

where:

- $\Pi(P_X, P_Y)$ - all possible joint distributions.

Wasserstein GAN

Earth-mover distance can be learned by:

$$\mathcal{L} = \mathop{\mathbb{E}}_{X \sim \mathrm{real}} f(x) - \mathop{\mathbb{E}}_{Z \sim \dots} f(G(Z)) \to \min$$

where:

- $\sup \|\nabla f\| < \mathrm{const.}$

discriminator: $$\mathop{\mathbb{E}}_{X \sim \mathrm{real}} f(X) - \mathop{\mathbb{E}}_{Z \sim \dots} f(D(Z)) \to \max;$$

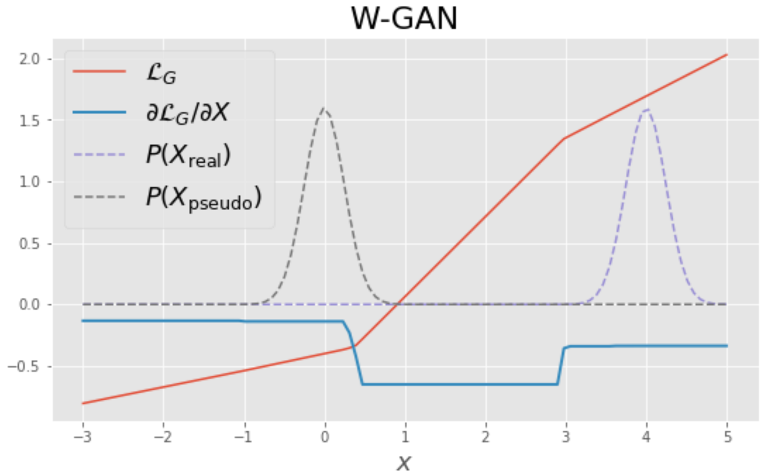
generator: $$\mathop{\mathbb{E}}_{Z \sim \dots} f(D(Z)) \to \max\,.$$

Wasserstein GAN

Restriction $\sup \|\nabla f\| < \mathrm{const}$ can be satisfied by clipping weights to a box.

```
def wgan_discriminator_update():
  params = params + lr * rmsprop_update()
  params = minimum(maximum(params, -c), c)
```

Wasserstein GAN



Discussion

Similar to EB-GAN:

- discriminator in W-GAN are trained as long as it is possible;
- also doesn't require any tricks to work;
- **always** provides proper gradients;

Clipping might be an optimization hell.

Summary

Summary

- classical GAN:
  - noise, regularization, ensemble etc;
- EB-GAN:
  - classificator $\to$ energy model;
  - non-vanishing gradients;
- W-GAN:
  - classificator $\to$ any function ($\sup \|\nabla f\| < \mathrm{const}$);
  - never vanishing gradients.

Bonus: domain adaptation

The idea

Credits to Ganin, Yaroslav, et al. "Domain-adversarial training of neural networks." Journal of Machine Learning Research 17.59 (2016): 1-35.

Setup:

- two domains $\mathcal{D}_1$ and $\mathcal{D}_2$;
- a problem over both domain;
- labels are available only for $\mathcal{D}_1$;
- difference between $\mathcal{D}_1$ and $\mathcal{D}_2$ does not interfere much with the problem;

Example:

- particle identification;
- domains: Monte-Carlo and real data;
- labels are available only for Monte-Carlo.

The idea

Lets break network f into two parts:

$$f(X) = h(g(X))$$

If

$$P(g(X) \mid \mathcal{D}_1) = P(g(X) \mid \mathcal{D}_2)$$

then the network's output is invariant to domain change.

The idea

$$\text{network}(x) = f(X) = h(g(X))$$

A discriminator network can serve as a proxy for distance:

$$\rho\left\{P(g(X) \mid \mathcal{D}_1), P(g(X) \mid \mathcal{D}_2)\right\} = \mathcal{L}_{\text{discriminator}} = \mathop{\mathbb{E}}_{X, y \sim \mathcal{D}} (d(g(X)), y) \to \min$$

Invariant features (parameters of g) can be learnt from discriminator d.

The math

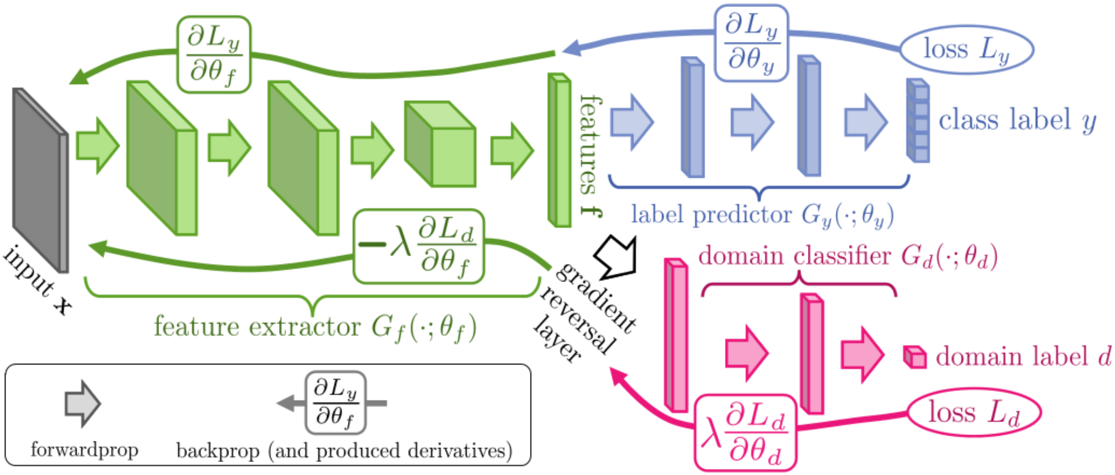
$$\mathcal{L} = \mathop{\mathbb{E}}_{X,Y \in \mathcal{D}_1} l(f(X), Y) - \lambda \left[\mathop{\mathbb{E}}_{X \in \mathcal{D}_1} l_d^+(d(g(X))) + \mathop{\mathbb{E}}_{X \in \mathcal{D}_2} l_d^-(d(g(X))) \right] =$$

$$\mathcal{L}_y + \mathcal{L}_d$$

$$\begin{aligned} h^*, g^* &= \arg\min_{h,g} \mathcal{L}; \\ d^* &= \arg\max_{d} \mathcal{L}; \end{aligned}$$

- l - loss for the problem (e.g. cross-entropy for classification);
- l_d - discriminative loss; $l_d^+(z) = l_d(z, y = 1), l_d^+(z) = l_d(z, y = 0)$

Domain Adaptation Network



Update rules

$$
\begin{aligned}
\theta_h &\leftarrow \theta_h - \alpha \frac{\partial \mathcal{L}_y}{\partial \theta_h}; \\
\theta_d &\leftarrow \theta_d - \alpha \frac{\partial \mathcal{L}_y}{\partial \theta_d}; \\
\theta_g &\leftarrow \theta_g - \alpha \left(\frac{\partial \mathcal{L}_y}{\partial \theta_g} - \lambda \frac{\mathcal{L}_d}{\partial \theta_g} \right).
\end{aligned}
$$

Summary

- discriminator as a measure of invariance;
- forcing lower layers of a target network to produce invariant tranformation;
- gradients of a trained discriminator to eliminate domain dependency in features.