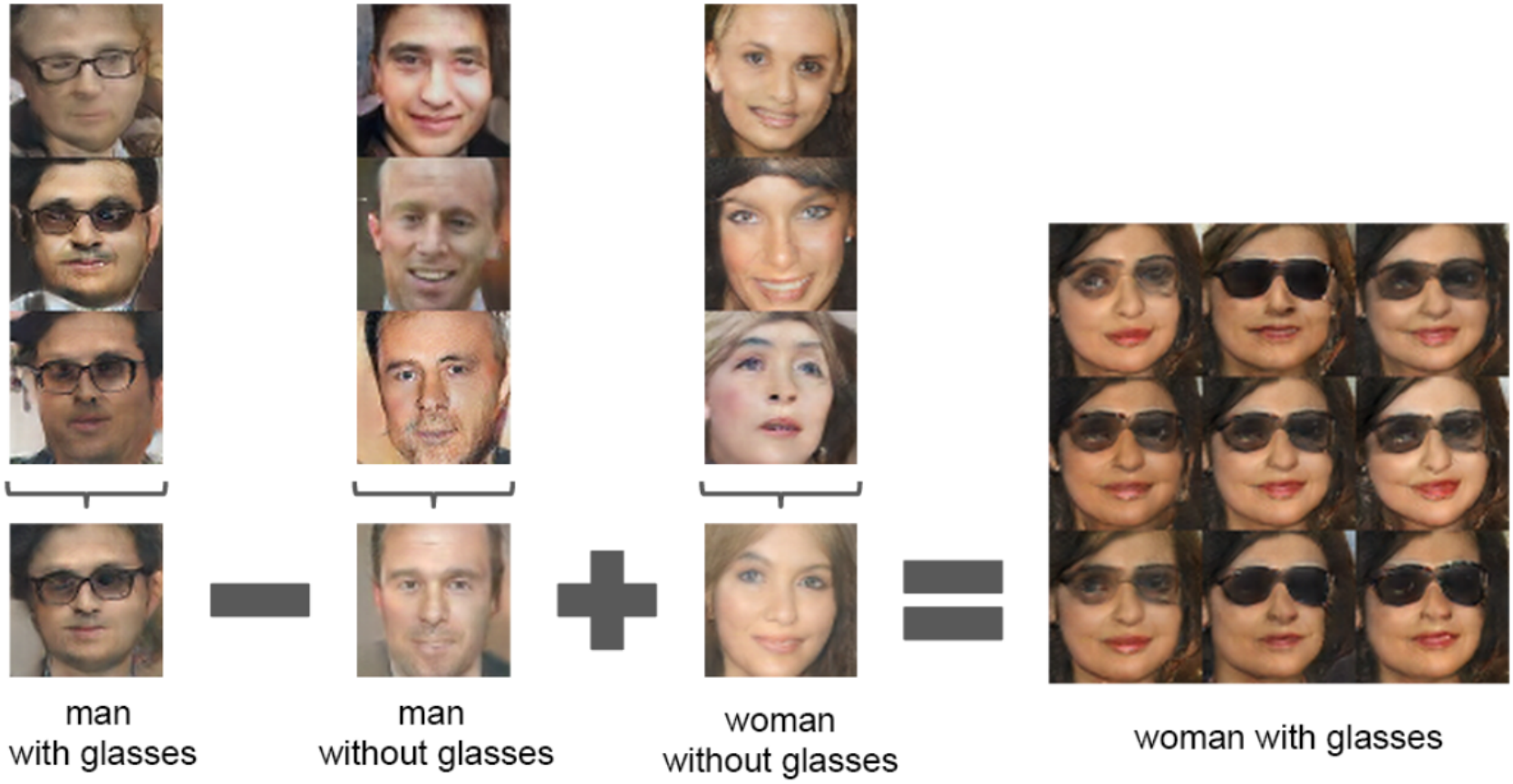
AdaGAN: Boosting Generative Models

Ilya Tolstikhin
ilya@tuebingen.mpg.de

joint work with Gelly[2], Bousquet[2], Simon-Gabriel[1], Schölkopf[1]

[1]MPI for Intelligent Systems [2]Google Brain



(Radford et al., 2015)

(Radford et al., 2015)

(Ngyuen et al., 2016)

(Youtube: Yota Ishida)

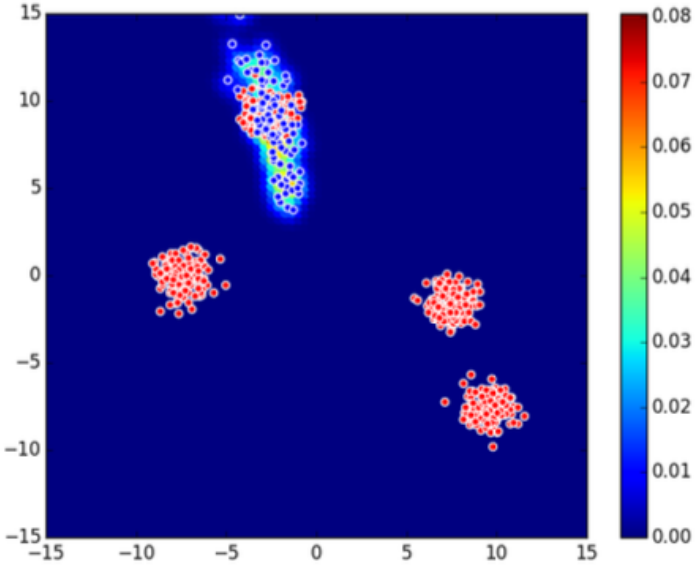
Motivation

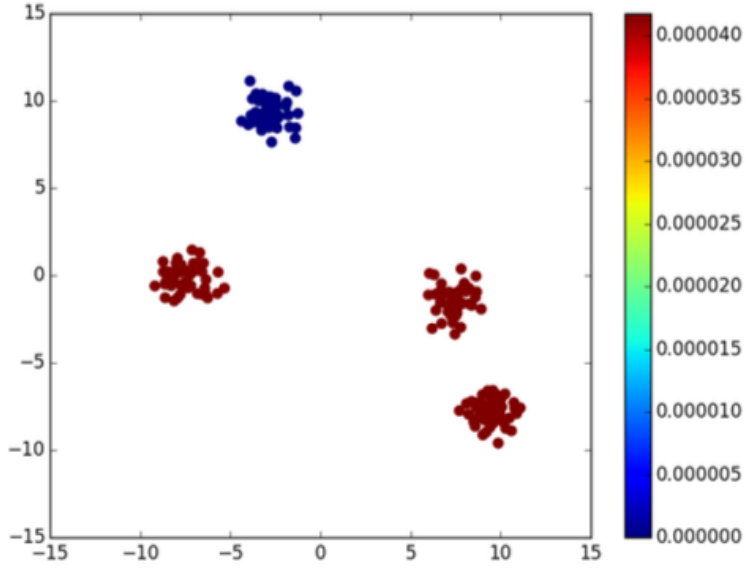
- Generative Adversarial Networks, Variational Autoencoders, and other unsupervised representation learning/generative modelling techniques ROCK!
- Dozens of papers being submitted, discussed, published, ...
- Among papers submitted to ICLR2017 **15** have titles matching 'Generative Adversarial*'.
- NIPS2016 Tutorial on GANs.
- GAN Workshop at NIPS2016 was the **second-largest**.

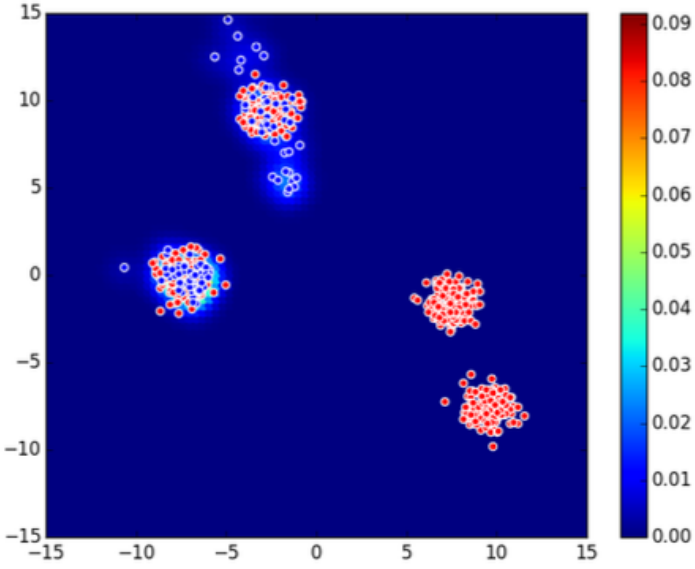
That's all very exciting. **However:**

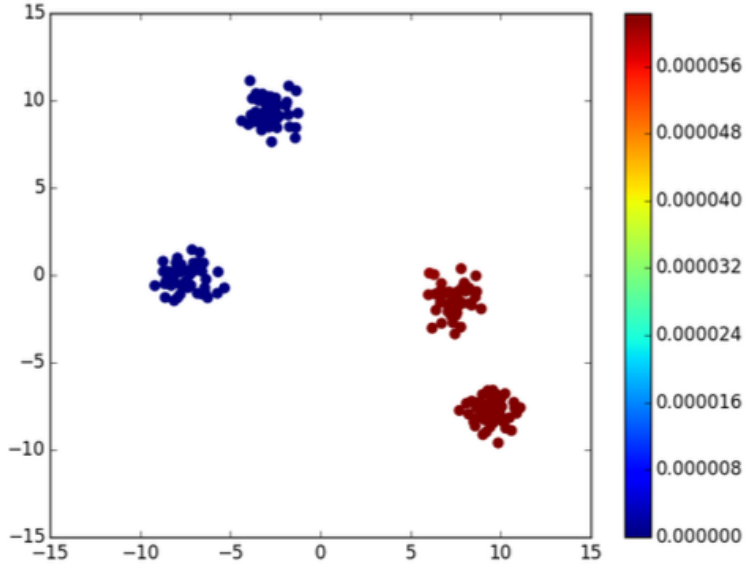
- GANs are **very hard to train**.
- The field is, apparently, in its "trials-and-errors" phase. People try all kinds of heuristic hacks. Sometimes they work, sometimes not...
- Very few papers manage to go beyond heuristics and provide a deeper conceptual understanding of what's going on.

AdaGAN: adaptive mixtures of GANs









AdaGAN: adaptive mixtures of GANs

Algorithm 1: AdaGAN, a meta-algorithm to construct a "strong" mixture of T individual GANs, trained sequentially. The mixture weight schedule ChooseMixtureWeight and the training set reweighting schedule UpdateTrainingWeights should be provided by the user. Section 3 gives a complete instance of this family.

Input: Training sample $S_N := \{X_1, \ldots, X_N\}$.

Output: Mixture generative model $G = G_T$.

Train vanilla GAN:

$W_1 = (1/N, \ldots, 1/N)$

$G_1 = \mathrm{GAN}(S_N, W_t)$

for $t = 2, \ldots, T$ **do**

#Choose a mixture weight for the next component

$\beta_t = \mathrm{ChooseMixtureWeight}(t)$

#Update weights of training examples

$W_t = \mathrm{UpdateTrainingWeights}(G_{t-1}, S_N, \beta_t)$

#Train t-th "weak" component generator G_t^c

$G_t^c = \mathrm{GAN}(S_N, W_t)$

#Update the overall generative model

#Notation below means forming a mixture of G_{t-1} *and* G_t^c.

$G_t = (1 - \beta_t) G_{t-1} + \beta_t G_t^c$

end for

Contents

1. **Generative Adversarial Networks and f-divergences**

   GANs are trying to minimize specific f-divergence

   $$\min_{P_{model} \in \mathcal{P}} D_f(P_{data}, P_{model}).$$

   At least, this is one way to explain what they are doing.

2. **Minimizing f-divergences with mixtures of distributions**

   We study theoretical properties of the following problem:

   $$\min_{\alpha_T, P_T \in \mathcal{P}} D_f \left(P_{data}, \sum_{t=1}^{T} \alpha_t P_t \right)$$

   and show that it can be reduced to

   $$\min_{P_T \in \mathcal{P}} D_f \left(\tilde{P}_{data}, P_T \right). \qquad (*)$$

3. **Back to GANs**

   Now we use GANs to train P_T in (*), which results in **AdaGAN**.

Contents

1. **Generative Adversarial Networks and f-divergences**

   GANs are trying to minimize specific f-divergence

   $$\min_{P_{model} \in \mathcal{P}} D_f(P_{data} \| P_{model}).$$

   At least, this is one way to explain what they are doing.

2. **Minimizing f-divergences with mixtures of distributions**

   We study theoretical properties of the following problem:

   $$\min_{\alpha_T, P_T \in \mathcal{P}} D_f \left(P_{data} \middle\| \sum_{t=1}^{T} \alpha_t P_t \right)$$

   and show that it can be reduced to

   $$\min_{P_T \in \mathcal{P}} D_f \left(\tilde{P}_{data} \| P_T \right). \qquad (*)$$

3. **Back to GANs**

   Now we use GANs to train P_T in (*), which results in **AdaGAN**.

f-GAN: Adversarial training for f-divergence minimization

For any probability distributions P and Q, f-divergence is

$$D_f(Q\|P) := \int f\left(\frac{dQ}{dP}(x)\right) dP(x),$$

where $f\colon (0,\infty) \to \mathbb{R}$ is convex and satisfies $f(1) = 0$.

Properties of f-divergence:

1. $D_f(P\|Q) \geq 0$ and $D_f(P\|Q) = 0$ when $P = Q$;
2. $D_f(P\|Q) = D_{f^\circ}(Q\|P)$ for $f^\circ(x) := xf(1/x)$;
3. $D_f(P\|Q)$ is symmetric when $f(x) = xf(1/x)$ for all x;
4. $D_f(P\|Q) = D_g(P\|Q)$ for any $g(x) = f(x) + C(x-1)$;
5. Taking $f_0(x) := f(x) - (x-1)f'(1)$ is a good choice. It is nonnegative, nonincreasing on $(0,1]$, and nondecreasing on $[1,\infty)$.

f-GAN: Adversarial training for f-divergence minimization

For any probability distributions P and Q, f-divergence is

$$D_f(Q\|P) := \int f\left(\frac{dQ}{dP}(x)\right) dP(x),$$

where $f\colon (0,\infty) \to \mathbb{R}$ is convex and satisfies $f(1) = 0$.

Examples of f-divergences:

1. Kullback-Leibler divergence for $f(x) = x \log x$;
2. Reverse Kullback-Leibler divergence for $f(x) = -\log x$;
3. Total Variation **distance** for $f(x) = |x - 1|$;
4. Jensen-Shannon divergence for $f(x) = -(x+1)\log\frac{x+1}{2} + x\log x$:

$$JS(P\|Q) = KL\left(P\middle\|\frac{P+Q}{2}\right) + KL\left(Q\middle\|\frac{P+Q}{2}\right).$$

f-GAN: Adversarial training for f-divergence minimization

Kullback-Leibler minimization: assume $X_1, \ldots, X_n \sim Q$

$$\begin{aligned}
\min_P \mathrm{KL}(Q\|P) \quad &\Leftrightarrow \quad \min_P \int \log\left(\frac{dQ}{dP}(x)\right) dQ(x) \\
&\Leftrightarrow \quad \min_P - \int \log(dP(x)) dQ(x) \\
&\Leftrightarrow \quad \max_P \int \log(dP(x)) dQ(x) \\
&\approx \quad \max_P \frac{1}{N} \sum_{i=1}^{N} \log(dP(X_i)).
\end{aligned}$$

Reverse Kullback-Leibler minimization:

$$\begin{aligned}
\min_P \mathrm{KL}(P\|Q) \quad &\Leftrightarrow \quad \min_P \int \log\left(\frac{dP}{dQ}(x)\right) dP(x) \\
&\Leftrightarrow \quad \min_P -H(P) - \int \log(dQ(x)) dP(x) \\
&\Leftrightarrow \quad \max_P \; H(P) + \int \log(dQ(x)) dP(x).
\end{aligned}$$

f-GAN: Adversarial training for f-divergence minimization

Now we will follow (Nowozin et al., NIPS2016).

For any probability distributions P and Q over $\mathcal{X}$, f-divergence is

$$D_f(Q\|P) := \int f\left(\frac{dQ}{dP}(x)\right) dP(x).$$

Variational representation: a very neat result, saying that

$$D_f(Q\|P) = \sup_{g\colon \mathcal{X}\to \mathrm{dom}(f^*)} \mathbb{E}_{X\sim Q}[g(X)] - \mathbb{E}_{Y\sim P}\big[f^*\big(g(Y)\big)\big],$$

where $f^*(x) := \sup_u x\cdot u - f(u)$ is a convex conjugate of f.

D_f can be computed by solving an optimization problem!

The f-divergence minimization can be now written as:

$$\inf_P D_f(Q\|P) \quad \Leftrightarrow \quad \inf_P \sup_{g\colon \ldots} \mathbb{E}_{X\sim Q}[g(X)] - \mathbb{E}_{Y\sim P}\big[f^*\big(g(Y)\big)\big].$$

f-GAN: Adversarial training for f-divergence minimization

The f-divergence minimization can be now written as:

$$\inf_P D_f(Q\|P) \quad \Leftrightarrow \quad \inf_P \sup_{g\colon \mathcal{X}\to \mathrm{dom}(f^*)} \mathbb{E}_{X\sim Q}[g(X)] - \mathbb{E}_{Y\sim P}\big[f^*\big(g(Y)\big)\big].$$

1. Take P to be a distribution of $G_\theta(Z)$, where $Z \sim P_Z$ is a latent noise and G_θ is a deep net;
2. JS divergence corresponds to $f(x) = -(x+1)\log\frac{x+1}{2} + x\log x$ and $f^*(t) = -\log\big(2 - e^t\big)$ and thus domain of f^* is $(-\infty, \log 2)$;
3. Take $g = g_f \circ D_\omega$, where $g_f(v) = \log 2 - \log(1 + e^{-v})$ and D_ω is a deep net (**check that** $g(x) < \log 2$ for all x).

Then up to additive $2\log 2$ term we get:

$$\inf_{G_\theta} \sup_{D_\omega} \mathbb{E}_{X\sim Q} \log \frac{1}{1+e^{-D_\omega(X)}} + \mathbb{E}_{Z\sim P_Z} \log\left(1 - \frac{1}{1+e^{-D_\omega\left(G_\theta(Z)\right)}}\right)$$

Compare to the original GAN objective (Goodfellow et al., NIPS2014)

$$\inf_{G_\theta} \sup_{D_\omega} \mathbb{E}_{X\sim P_d}[\log D_\omega(X)] + \mathbb{E}_{Z\sim P_Z}[\log\big(1 - D_\omega(G_\theta(Z))\big)].$$

f-GAN: Adversarial training for f-divergence minimization

Then we get the original GAN objective (Goodfellow et al., NIPS2014)

$$\inf_{G_\theta} \sup_{D_\omega} \mathbb{E}_{X\sim P_d}[\log D_\omega(X)] + \mathbb{E}_{Z\sim P_Z}[\log\big(1 - D_\omega(G_\theta(Z))\big)]. \qquad (*)$$

Let's denote a distribution of $G_\theta(Z)$ when $Z \sim P_Z$ with P_θ.

- For any fixed generator G_θ, the theoretical-optimal D^* is:

$$D^*_\theta(x) = \frac{dP_d(x)}{dP_d(x) + dP_\theta(x)}.$$

- If we insert this back to (*) we get

$$\inf_{G_\theta} 2 \cdot JS(P_d \| P_\theta) - \log(4).$$

f-GAN: Adversarial training for f-divergence minimization

The f-divergence minimization can be now written as:

$$\inf_P D_f(Q\|P) \quad \Leftrightarrow \quad \inf_P \sup_{g\colon \mathcal{X}\to \mathrm{dom}(f^*)} \mathbb{E}_{X\sim Q}[g(X)] - \mathbb{E}_{Y\sim P}\big[f^*\big(g(Y)\big)\big].$$

This is GREAT, because:

1. Estimate $\mathbb{E}_{X\sim Q}[g(X)]$ using an i.i.d. sample (data) $X_1, \ldots, X_N \sim Q$:

$$\mathbb{E}_{X\sim Q}[g(X)] \approx \frac{1}{N}\sum_{i=1}^{N} g(X_i);$$

2. Often we don't know the analytical form of P, but can only sample from P. No problems! Sample i.i.d. $Y_1, \ldots, Y_M \sim P$ and write:

$$\mathbb{E}_{Y\sim P}\big[f^*\big(g(Y)\big)\big] \approx \frac{1}{M}\sum_{j=1}^{M} f^*\big(g(Y_j)\big).$$

This results in implicit generative modelling: using complex model distributions, which are analytically intractable, but easy to sample.

f-GAN: Adversarial training for f-divergence minimization

Note that there is still a difference between doing GAN:

$$\inf_{G_\theta} \sup_{D_\omega} \mathbb{E}_{X\sim P_d}[\log D_\omega(X)] + \mathbb{E}_{Z\sim P_Z}[\log\big(1 - D_\omega(G_\theta(Z))\big)]$$

and minimizing $\mathrm{JS}(P_{data}, P_{model})$:

$$\inf_{P_{model}} \sup_{g\colon \ldots} \mathbb{E}_{X\sim P_{data}}[g(X)] - \mathbb{E}_{Y\sim P_{model}}\left[f^*_{JS}\big(g(Y)\big)\right],$$

because

1. We replaced expectations with sample averages. Uniform lows of large numbers may not apply, as our function classes are huge;
2. Instead of taking supremum over all possible witness functions g we optimize over restricted class of DNNs (which is still huge);
3. In practice we never optimize g/D_ω "to the end". This leads to vanishing gradients and other bad things.

Contents

1. **Generative Adversarial Networks and f-divergences**

   GANs are trying to minimize specific f-divergence

   $$\min_{P_{model} \in \mathcal{P}} D_f(P_{data} \| P_{model}).$$

   At least, this is one way to explain what they are doing.

2. **Minimizing f-divergences with mixtures of distributions**

   We study theoretical properties of the following problem:

   $$\min_{\alpha_T, P_T \in \mathcal{P}} D_f \left(P_{data} \middle\| \sum_{t=1}^{T} \alpha_t P_t \right)$$

   and show that it can be reduced to

   $$\min_{P_T \in \mathcal{P}} D_f \left(\tilde{P}_{data} \| P_T \right). \qquad (*)$$

3. **Back to GANs**

   Now we use GANs to train P_T in (*), which results in **AdaGAN**.

Minimizing f-divergences with mixtures of distributions

Assumption: Let's assume we know more or less how to solve

$$\min_{Q \in \mathcal{G}} D_f(Q \,\|\, P)$$

for any f and any P, given we have an i.i.d. sample from P.

New problem: Now we want to approximate an unknown P_d in D_f using a mixture model of the form

$$P^T_{model} := \sum_{i=1}^{T} \alpha_i P_i.$$

Greedy strategy:

1. Train P_1 to minimize $D_f(P_1 \| P_d)$, set $\alpha_1 = 1$, and get $P^1_{model} = P_1$;
2. Assume we trained $P_1, \ldots, P_t$ with weights $\alpha_1, \ldots, \alpha_t$. We are going to train one more component Q, choose weight β and update

$$P^{t+1}_{model} = \sum_{i=1}^{t} (1-\beta)\alpha_i P_i + \beta Q.$$

Minimizing f-divergences with mixtures of distributions

1. Train P_1 to minimize $D_f(P_1 \| P_d)$, set $\alpha_1 = 1$, and get $P^1_{model} = P_1$;
2. Assume we trained $P_1, \ldots, P_t$ with weights $\alpha_1, \ldots, \alpha_t$. We are going to train one more component Q, choose weight β and update

$$P^{t+1}_{model} = \sum_{i=1}^{t} (1-\beta)\alpha_i P_i + \beta Q.$$

The goal: we want to choose $\beta \in [0, 1]$ and Q greedily, so as to minimize

$$D_f((1-\beta)P_g + \beta Q \,\|\, P_d),$$

where we denoted $P_g := P^t_{model}$.

A modest goal: find $\beta \in [0, 1]$ and Q, such that, for $c < 1$:

$$D_f((1-\beta)P_g + \beta Q \,\|\, P_d) \leq c \cdot D_f(P_g \,\|\, P_d).$$

Inefficient in practice: as $t \to \infty$ we expect $\beta \to 0$. Only β fraction of points gets sampled from $Q \Rightarrow$ gets harder and harder to tune Q.

Minimizing f-divergences with mixtures of distributions

$$D_f((1-\beta)P_g + \beta Q \,\|\, P_d) \qquad (*)$$

Inefficient in practice: as $t \to \infty$ we expect $\beta \to 0$. Only β fraction of points gets sampled from $Q \Rightarrow$ gets harder and harder to tune Q.

Workaround: instead optimize an upper bound on (*), which contains a term $D_f(Q, Q_0)$ for some Q_0:

Lemma Given P_d, P_g, and $\beta \in [0,1]$, for any Q and any R with $\beta\, dR \le dP_d$

$$(*) \le \beta D(Q \,\|\, R) + (1-\beta) D_f\left(P_g \,\|\, \frac{P_d - \beta R}{1-\beta}\right).$$

If furthermore D_f is a Hilbertian metric, then, for any R, we have

$$\sqrt{(*)} \le \sqrt{\beta D_f(Q \,\|\, R)} + \sqrt{D_f\left((1-\beta)P_g + \beta R \,\|\, P_d\right)}.$$

JS-divergence, TV, Hellinger, and others are Hilbertian metrics.

Minimizing f-divergences with mixtures of distributions

$$D_f((1-\beta)P_g + \beta Q \,\|\, P_d) \qquad (*)$$

Lemma Given P_d, P_g, and $\beta \in [0,1]$, for any Q and any R with $\beta\, dR \le dP_d$

$$(*) \le \beta D(Q \,\|\, R) + (1-\beta) D_f\left(P_g \,\|\, \frac{P_d - \beta R}{1-\beta}\right).$$

If furthermore D_f is a Hilbertian metric, then, for any R, we have

$$(*) \le \left(\sqrt{\beta D_f(Q \,\|\, R)} + \sqrt{D_f\left((1-\beta)P_g + \beta R \,\|\, P_d\right)}\right)^2.$$

Recipe: choose R^* to minimize the right-most terms and then minimize $D_f(Q\|R^*)$, which we can do given we have an i.i.d. sample from R^*.

Minimizing f-divergences with mixtures of distributions

First surrogate upper bound

Lemma Given P_d, P_g, and $\beta \in [0,1]$, for any Q and R, if D_f is Hilbertian

$$(*) \leq \left(\sqrt{\beta D_f(Q \,\|\, R)} + \sqrt{D_f\left((1-\beta)P_g + \beta R \,\|\, P_d\right)} \right)^2 .$$

The optimal R^* is given in the next result:

Theorem Given P_d, P_g, and $\beta \in (0,1]$, the solution to

$$\min_{R \in \mathbb{P}} D_f((1-\beta)P_g + \beta R \,\|\, P_d),$$

where $\mathbb{P}$ is a class of all probability distributions, has the density

$$dR^*_\beta(x) = \frac{1}{\beta}\left(\lambda^* dP_d(x) - (1-\beta)dP_g(x)\right)_+$$

for some unique λ^* satisfying $\int dR^*_\beta = 1$.
Also, $\lambda^* = 1$ if and only if $P_d((1-\beta)dP_g > dP_d) = 0$, which is equivalent to $\beta dQ^*_\beta = dP_d - (1-\beta)dP_g$.

Minimizing f-divergences with mixtures of distributions

Second surrogate upper bound

Lemma Given P_d, P_g, and $\beta \in [0,1]$, for any Q and any R with $\beta\, dR \le dP_d$

$$D_f((1-\beta)P_g + \beta Q \,\|\, P_d) \le \beta D(Q \,\|\, R) + (1-\beta) D_f\left(P_g \,\|\, \frac{P_d - \beta R}{1-\beta}\right).$$

The optimal R^* is given in the next result:

Theorem Given P_d, P_g, and $\beta \in (0,1]$, assume $P_d\,(dP_g = 0) < \beta$. Then the solution to

$$\min_{R:\beta dR \le dP_d} D_f\left(P_g \,\|\, \frac{P_d - \beta R}{1-\beta}\right)$$

is given by the density

$$dR_\beta^\dagger(x) = \frac{1}{\beta}\left(dP_d(x) - \lambda^\dagger(1-\beta)dP_g(x)\right)_+$$

for a unique $\lambda^\dagger \ge 1$ satisfying $\int dR_\beta^\dagger = 1$.

Minimizing f-divergences with mixtures of distributions

Two different optimal R^*:

$$dR^*_\beta(x) = \frac{1}{\beta}\left(\lambda^* dP_d(x) - (1-\beta)dP_g(x)\right)_+$$

$$dR^\dagger_\beta(x) = \frac{1}{\beta}\left(dP_d(x) - \lambda^\dagger(1-\beta)dP_g(x)\right)_+$$

- Very symmetrically looking, but no obvious connections found...
- Both do not depend on f!
- λ^* and $\lambda^\dagger$ are implicitly defined via fixed-point equations.
- $dR^*_\beta(x)$ is also a solution to the original problem, that is of

$$\min_{Q\in\mathbb{R}} D_f((1-\beta)P_g + \beta Q \,\|\, P_d).$$

We see that setting $\beta = 1$ is the best possible choice in theory. So we'll have to use various heuristics, including uniform, constant, ...

Minimizing f-divergences with mixtures of distributions

$$D_f((1-\beta)P_g + \beta Q \,\|\, P_d) \qquad (*)$$

Lemma Given P_d, P_g, and $\beta \in [0,1]$, for any Q and any R with $\beta\, dR \le dP_d$

$$(*) \le \beta D(Q \,\|\, R) + (1-\beta) D_f\left(P_g \,\|\, \frac{P_d - \beta R}{1-\beta}\right).$$

If furthermore D_f is a Hilbertian metric, then, for any R, we have

$$(*) \le \left(\sqrt{\beta D_f(Q \,\|\, R)} + \sqrt{D_f\left((1-\beta)P_g + \beta R \,\|\, P_d\right)}\right)^2.$$

Recipe: choose R^* to minimize the right-most terms and then minimize $D_f(Q\|R^*)$, which we can do given we have an i.i.d. sample from R^*.

Minimizing f-divergences with mixtures of distributions

Recipe: choose R^* to minimize the right-most terms in the surrogate upper bounds and then minimize the first term $D_f(Q\|R^*)$.

- We found optimal R^* distributions. Let's now assume we are super powerful and managed to find Q with $D_f(Q\|R^*) = 0$.
- In other words, we are going to update our model:

$$P_g \quad \Rightarrow \quad (1-\beta)P_g + \beta Q$$

  with Q either R^*_β or $R^\dagger_\beta$.
- How well are we doing in terms of the original objective

$$D_f((1-\beta)P_g + \beta Q \,\|\, P_d)?$$

Theorem We have:

$$D_f\big((1-\beta)P_g+\beta R^*_\beta \,\big\|\, P_d\big) \le D_f\big((1-\beta)P_g+\beta P_d \,\big\|\, P_d\big) \le (1-\beta)D_f(P_g \,\|\, P_d)$$

and

$$D_f\big((1-\beta)P_g + \beta R^\dagger_\beta \,\big\|\, P_d\big) \le (1-\beta)D_f(P_g \,\|\, P_d).$$

Minimizing f-divergences with mixtures of distributions

Other highlights:

- More careful convergence analysis. In particular we provide a necessary and sufficient condition for the iterative process to converge in a finite number of steps: there exists $M > 0$ such that

$$P_d((1-\beta)dP_1 > MdP_d) = 0.$$

In other words, $P_1 \ll P_d$

- In practice we don't attain $D_f(Q\|R^*) = 0$. We prove that given

$$D_f(Q\|R^*_\beta) \leq \gamma D_d(P_g\|P_d) \quad \text{or} \quad D_f(Q\|R^\dagger_\beta) \leq \gamma D_d(P_g\|P_d)$$

we achieve

$$D_f((1-\beta)P_g + \beta Q \,\|\, P_d) \leq c \cdot D_f(P_g \,\|\, P_d)$$

with $c = c(\gamma, \beta) < 1$. This is kind of similar to "weak learnability" in AdaBoost.

Contents

1. **Generative Adversarial Networks and f-divergences**

   GANs are trying to minimize specific f-divergence

   $$\min_{P_{model} \in \mathcal{P}} D_f(P_{data} \| P_{model}).$$

   At least, this is one way to explain what they are doing.

2. **Minimizing f-divergences with mixtures of distributions**

   We study theoretical properties of the following problem:

   $$\min_{\alpha_T, P_T \in \mathcal{P}} D_f \left(P_{data} \middle\| \sum_{t=1}^{T} \alpha_t P_t \right)$$

   and show that it can be reduced to

   $$\min_{P_T \in \mathcal{P}} D_f \left(\tilde{P}_{data} \| P_T \right). \qquad (*)$$

3. **Back to GANs**

   Now we use GANs to train P_T in (*), which results in **AdaGAN**.

AdaGAN: adaptive mixtures of GANs

- We want to find Q so as to minimize $D_f(Q\|R^*_\beta)$, where

$$dR^*_\beta(x) = \frac{1}{\beta}\left(\lambda^* dP_d(x) - (1-\beta)dP_g(x)\right)_+ .$$

- We would like to use GAN for that. How do we sample from R^*_β?
- Let's rewrite:

$$dR^*_\beta(x) = \frac{dP_d(x)}{\beta}\left(\lambda^* - (1-\beta)\frac{dP_g}{dP_d}(x)\right)_+ .$$

- Recall: $\mathbb{E}_{P_d}[\log D(X)] + \mathbb{E}_{P_g}[\log\big(1 - D(Y)\big)]$ is maximized by:

$$D^*(x) = \frac{dP_d(x)}{dP_d(x) + dP_g(x)} \quad \Leftrightarrow \quad \frac{dP_g}{dP_d}(x) = \frac{1 - D^*(x)}{D^*(x)} := h(x).$$

- Each training sample end up with a weight

$$w_i = \frac{p_i}{\beta}\big(\lambda^* - (1-\beta)h(x_i)\big)_+ \approx \frac{1}{N\beta}\big(\lambda^* - (1-\beta)h(x_i)\big)_+ . \quad (**)$$

- Finally, we compute λ^* using (**).

AdaGAN: adaptive mixtures of GANs

Each training sample end up with a weight

$$w_i = \frac{p_i}{\beta}\big(\lambda^* - (1-\beta)h(x_i)\big)_+ \approx \frac{1}{N\beta}\left(\lambda^* - (1-\beta)\frac{1-D(x_i)}{D(x_i)}\right)_+.$$

Now we run a usual GAN with an importance-weighted sample.

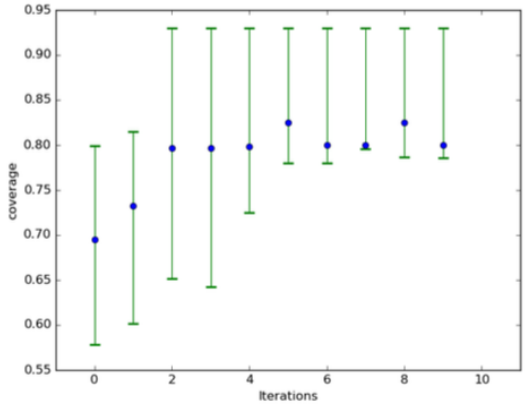
- $D(x_i) \approx 1$: our current model P_g is "missing" this point. Big w_i.
- $D(x_i) \approx 0$: x_i is covered by P_g nicely. Zero w_i.
- The better our model gets, the more data points will be zeroed.
- This leads to an aggressive adaptive iterative procedure.
- β can be also chosen so as to force $\#\{w_i : w_i > 0\} \approx r \cdot N$.

AdaGAN: adaptive mixtures of GANs

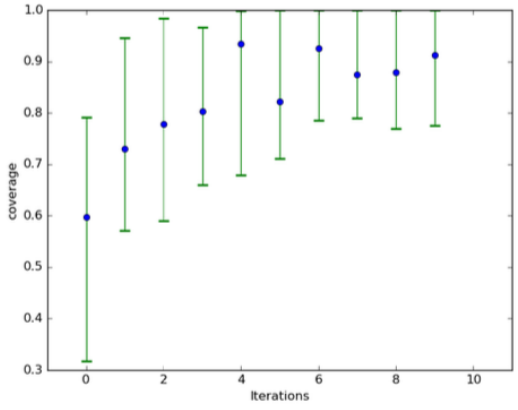
Coverage metric C: mass of the true data "covered" by the model:

$$C := P_{data}(dP_{model}(X) > t) \quad \text{where} \quad P_{model}(dP_{model} > t) = 0.95.$$

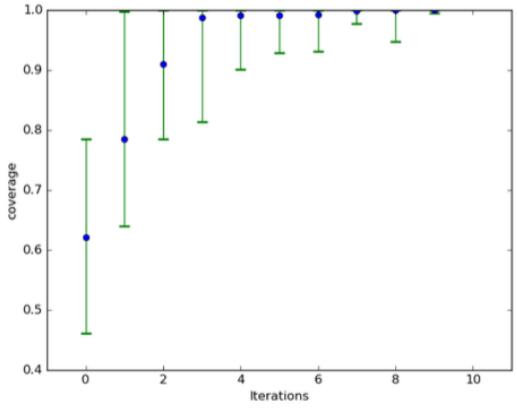
Experimental setting: Mixture of 5 two-dimensional Gaussians.



"Best of T GANs"



"Bagging"



"AdaGAN'

Blue points: median over 35 random runs;

Green error bars: 90% intervals. Not the confidence intervals for medians!

AdaGAN: adaptive mixtures of GANs

Coverage metric C: mass of the true data "covered" by the model:

$$C := P_{data}(dP_{model}(X) > t) \quad \text{where} \quad P_{model}(dP_{model} > t) = 0.95.$$

	Modes : 1	*Modes* : 2	*Modes* : 3	*Modes* : 5	*Modes* : 7	*Modes* : 10
Vanilla	0.97 (0.9; 1.0)	0.88 (0.4; 1.0)	0.63 (0.5; 1.0)	0.72 (0.5; 0.8)	0.58 (0.4; 0.8)	0.59 (0.2; 0.7)
Best of T (T=3)	0.99 (1.0; 1.0)	0.96 (0.9; 1.0)	0.91 (0.7; 1.0)	0.80 (0.7; 0.9)	0.84 (0.7; 0.9)	0.70 (0.6; 0.8)
Best of T (T=10)	0.99 (1.0; 1.0)	0.99 (1.0; 1.0)	0.98 (0.8; 1.0)	0.80 (0.8; 0.9)	0.87 (0.8; 0.9)	0.71 (0.7; 0.8)
Ensemble (T=3)	0.99 (1.0; 1.0)	0.98 (0.9; 1.0)	0.93 (0.8; 1.0)	0.78 (0.6; 1.0)	0.85 (0.6; 1.0)	0.80 (0.6; 1.0)
Ensemble (T=10)	1.00 (1.0; 1.0)	0.99 (1.0; 1.0)	1.00 (1.0; 1.0)	0.91 (0.8; 1.0)	0.88 (0.8; 1.0)	0.89 (0.7; 1.0)
TopKLast0.5 (T=3)	0.98 (0.9; 1.0)	0.98 (0.9; 1.0)	0.95 (0.9; 1.0)	0.95 (0.8; 1.0)	0.86 (0.7; 1.0)	0.86 (0.6; 0.9)
TopKLast0.5 (T=10)	0.99 (1.0; 1.0)	0.98 (0.9; 1.0)	0.98 (1.0; 1.0)	0.99 (0.8; 1.0)	0.99 (0.8; 1.0)	1.00 (0.8; 1.0)
Boosted (T=3)	0.99 (1.0; 1.0)	0.99 (0.9; 1.0)	0.98 (0.9; 1.0)	0.91 (0.8; 1.0)	0.91 (0.8; 1.0)	0.86 (0.7; 1.0)
Boosted (T=10)	1.00 (1.0; 1.0)	1.00 (1.0; 1.0)	1.00 (1.0; 1.0)	1.00 (1.0; 1.0)	1.00 (1.0; 1.0)	1.00 (1.0; 1.0)

TopKLast r: β_t is chosen to zero out $r \cdot N$ of the training;

Boosted: AdaGAN with $\beta_t = 1/t$.

AdaGAN: adaptive mixtures of GANs

Coverage metric C: mass of the true data "covered" by the model:

$$C := P_{data}(dP_{model}(X) > t) \quad \text{where} \quad P_{model}(dP_{model} > t) = 0.95.$$

Experimental setting: Mixture of 10 two-dimensional Gaussians.

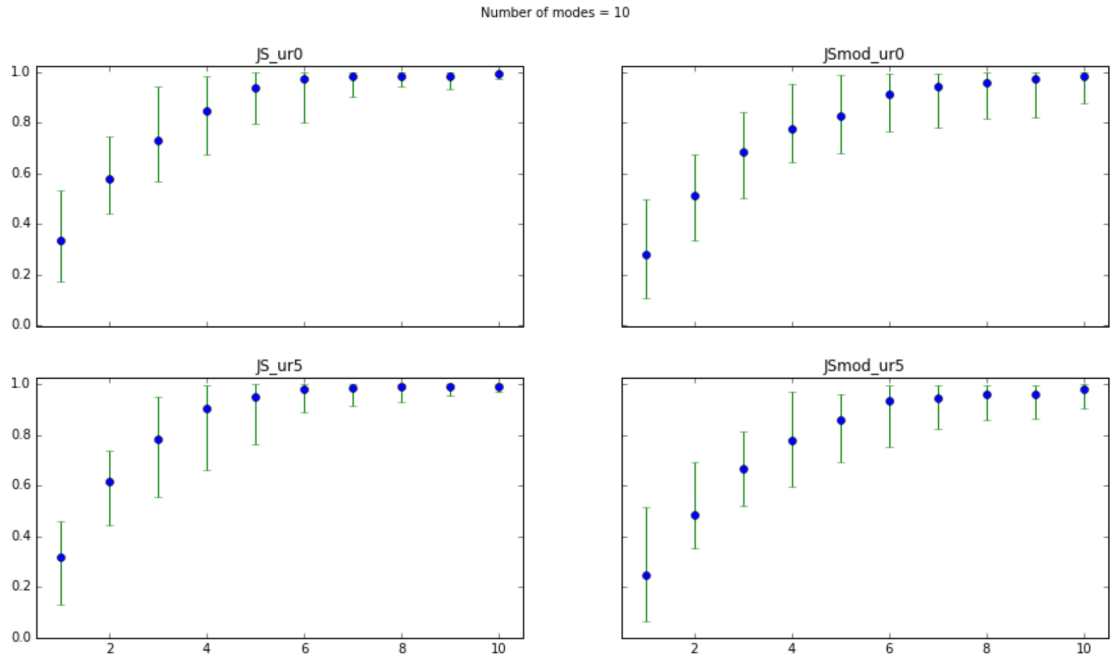


Figure 4: Comparison of AdaGAN ran with a GAN (top row) and with an unrolled GAN [4] (bottom). Coverage C of the true data by the model distribution P^T_{model}, as a function of iterations T. Experiments are similar to those of Figure 3, but with 10 modes. Top and bottom rows correspond to the usual and the unrolled GAN (with 5 unrolling steps) respectively, trained with the Jensen-Shannon objective (2) on the left, and with the modified objective originally proposed by [1] on the right. In terms of computation time, one step of AdaGAN with unrolled GAN corresponds to roughly 3 steps of AdaGAN with a usual GAN. On all images $T = 1$ corresponds to vanilla unrolled GAN.

AdaGAN: adaptive mixtures of GANs

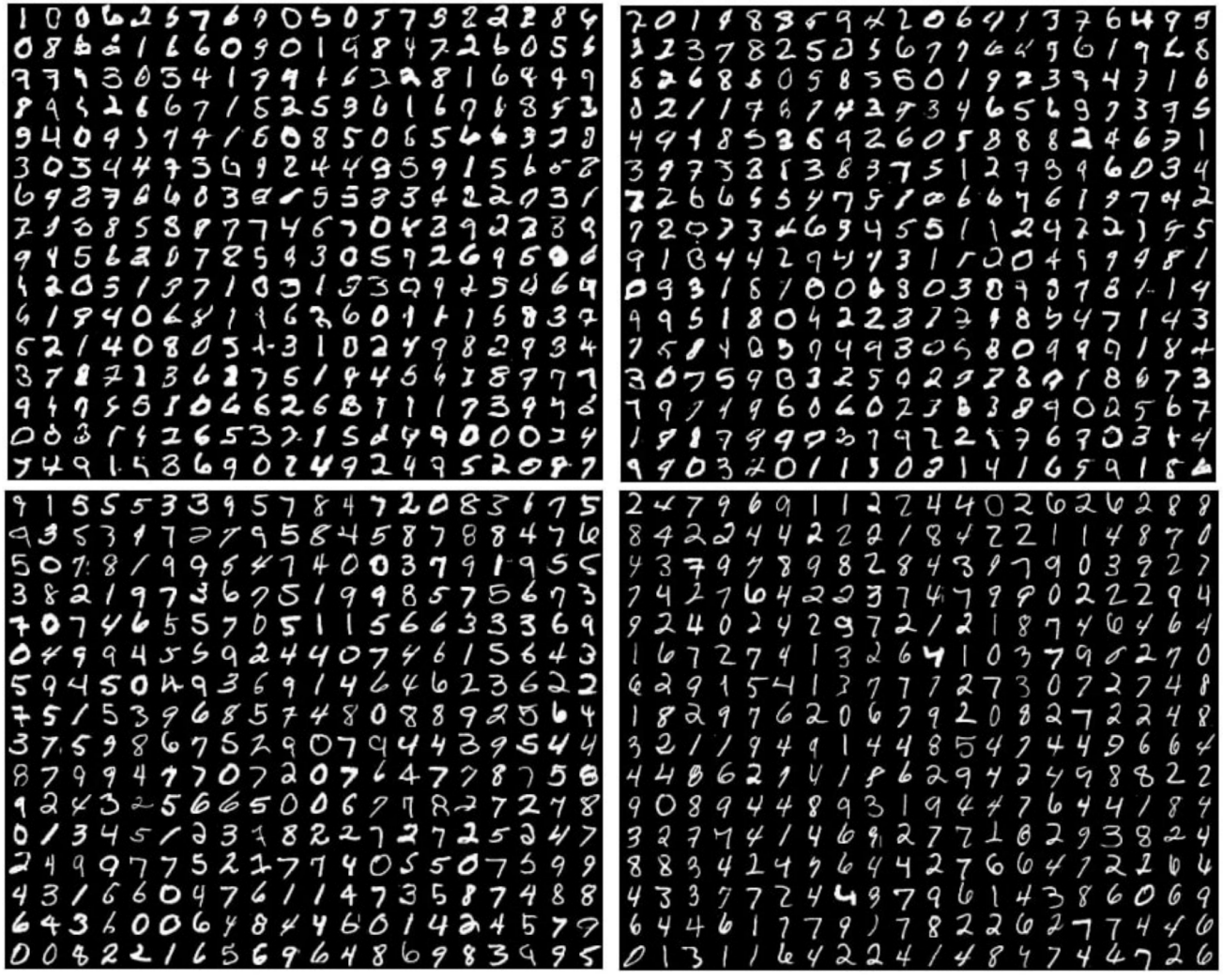


Figure 5: AdaGAN on MNIST. Bottom row are true MNIST digits with smallest (left) and highest (right) weights after re-weighting at the end of the first AdaGAN step. Those with small weight are thick and resemble those generated by the GAN after the first AdaGAN step (top left). After training with the re-weighted dataset during the second iteration of AdaGAN, the new mixture produces more thin digits (top right).

Conclusions

Pros:

- It seems that AdaGAN is solving the missing modes problem;
- AdaGAN is compatible with **any** implementation of GAN and more generally with any other implicit generative modelling techniques;
- Still easy to sample from;
- Easy to implement and great freedom with heuristics;
- Theoretically grounded!

Cons:

- We loose the nice "manifold" structure in the latent space;
- Increased computational complexity;
- Individual GANs are still very unstable.

Thank you!

AdaGAN: Boosting Generative Models. *Tolstikhin, Gelly, Bousquet, Simon-Gabriel, Schoelkopf*, NIPS 2017.

`https://github.com/tolstikhin/adagan`

Minimizing the optimal transport

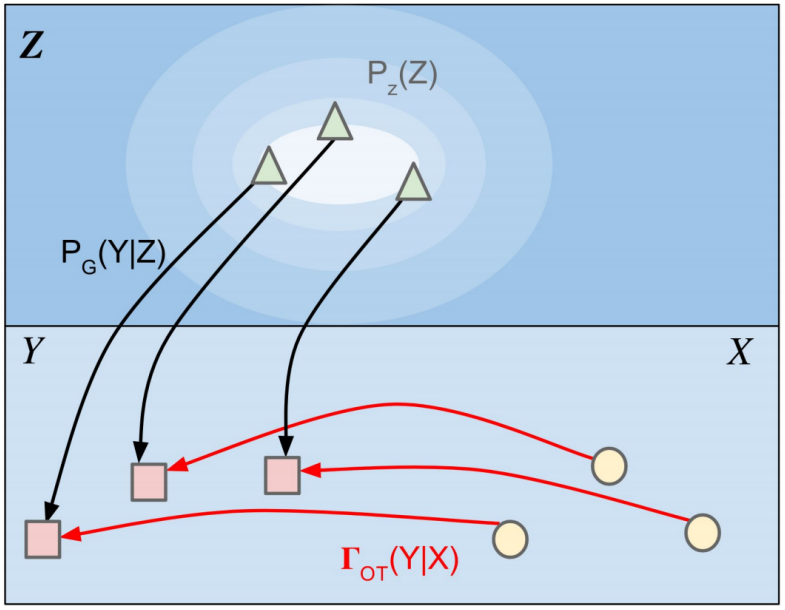
Optimal transport for a cost function $c(x, y) \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ is

$$W_c(P_X, P_G) := \inf_{\Gamma \in \mathcal{P}(X \sim P_X, Y \sim P_G)} \mathbb{E}_{(X,Y)\sim\Gamma}[c(X, Y)],$$

If $P_G(Y|Z = z) = \delta_{G(z)}$ for all $z \in \mathcal{Z}$, where $G \colon \mathcal{Z} \to \mathcal{X}$, we have

$$W_c(P_X, P_G) = \inf_{Q \colon Q_Z = P_Z} \mathbb{E}_{P_X} \mathbb{E}_{Q(Z|X)} \big[c\big(X, G(Z)\big)\big],$$

where Q_Z is the marginal distribution of Z when $X \sim P_X$, $Z \sim Q(Z|X)$.



Minimizing the optimal transport

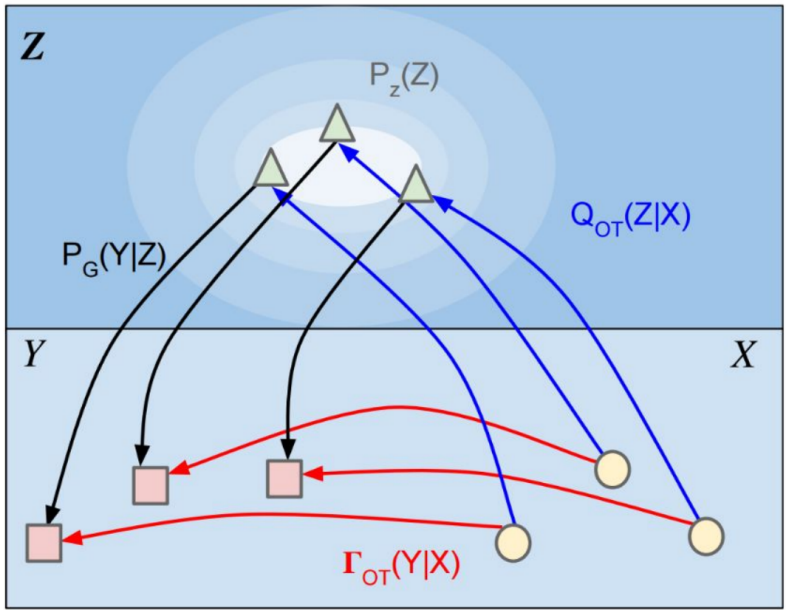
Optimal transport for a cost function $c(x, y)\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$ is

$$W_c(P_X, P_G) := \inf_{\Gamma \in \mathcal{P}(X \sim P_X, Y \sim P_G)} \mathbb{E}_{(X,Y)\sim\Gamma}[c(X, Y)],$$

If $P_G(Y|Z = z) = \delta_{G(z)}$ for all $z \in \mathcal{Z}$, where $G\colon \mathcal{Z} \to \mathcal{X}$, we have

$$W_c(P_X, P_G) = \inf_{Q\colon\, Q_Z = P_Z} \mathbb{E}_{P_X} \mathbb{E}_{Q(Z|X)} \big[c\big(X, G(Z)\big)\big],$$

where Q_Z is the marginal distribution of Z when $X \sim P_X$, $Z \sim Q(Z|X)$.



Relaxing the constraint

$$W_c(P_X, P_G) = \inf_{Q:\, Q_Z = P_Z} \mathbb{E}_{P_X} \mathbb{E}_{Q(Z|X)} \big[c\big(X, G(Z)\big) \big],$$

Penalized Optimal Transport replaces the constraint with a penalty:

$$\mathrm{POT}(P_X, P_G) := \inf_{Q} \mathbb{E}_{P_X} \mathbb{E}_{Q(Z|X)} \big[c\big(X, G(Z)\big) \big] + \lambda \cdot D(Q_Z, P_Z)$$

and uses the adversarial training in the $\mathcal{Z}$ space to approximate D.

- For the 2-Wasserstein distance $c(X, Y) = \|X - Y\|_2^2$ POT recovers Adversarial Auto-Encoders (Makhzani et al., 2016)
- For the 1-Wasserstein distance $c(X, Y) = \|X - Y\|_2$ POT and WGAN are solving the same problem from the primal and dual forms respectively.
- Importantly, unlike VAE, POT does not force $Q(Z|X = x)$ to intersect for different x, which is known to lead to the blurriness.

(Bousquet et al., 2017)

Further details on GAN: "the log trick"

$$\inf_{G_\theta} \sup_{T_\omega} \mathbb{E}_{X\sim P_d}[\log T_\omega(X)] + \mathbb{E}_{Z\sim P_Z}[\log\big(1 - T_\omega(G_\theta(Z))\big)]$$

In practice (verify this!) G_θ is often trained to instead minimize

$$-\mathbb{E}_{Z\sim P_Z}[\log\big(T_\omega(G_\theta(Z))\big)] \qquad \text{("The log trick")}$$

One possible explanation: For any f_r and P_G the optimal T^* in the variational representation of $D_{f_r}(P_X \| P_G)$ has the form

$$T^*(x) := f_r'\left(\frac{p_X(x)}{p_G(x)}\right) \quad \Leftrightarrow \quad \frac{p_X(x)}{p_G(x)} = (f_r')^{-1}(T^*(x))$$

1. Approximate $T^* \approx T_{\omega*}$ using the adversarial training with f_r;
2. Express $\frac{p_X}{p_G}(x) = (f_r')^{-1}(T_{w*}(x))$;
3. Approximate the desired f-divergence $D_f(P_X \| P_G)$ using

$$\int_{\mathcal{X}} f\left(\frac{p_X(x)}{p_G(x)}\right) p_G(x)dx \approx \int_{\mathcal{X}} f\left((f_r')^{-1}(T^*(x))\right) p_G(x)dx$$

Applying this argument with the JS f_r and $f(x) = \log(1 + x^{-1})$ recovers the log trick.

The corresponding f-divergence is much more mode-seeking than the reverse KL divergence.