Towards Principled Methods for Training Generative Adversarial Networks

Wasserstein GAN

KEISUKE FUKUTA

論文情報

Towards Principled Methods for Training Generative Adversarial Networks

- ICLR 2017 採択論文
- 数学強者がGANを解析してみましたという論文

Wasserstein GAN

- 3人中2人↑と同じ著者
- 解析した知見を利用して実用的なGANを提案

生成モデル

- 未知のデータ分布$P_r(x)$とモデルの分布$P_g(x)$を近づける

- よくある方法では、**尤度の最大化**

  ≅ **Kullback-Leibler (KL) divergence の最小化**

VAEではガウス分布を仮定しているのでKLが具体的に計算可能

GANでは分布を仮定していない

Kullback-Leibler (KL) divergence

$$KL(P_r||P_g) = \int_x P_r(x)\, log\, \frac{P_r(x)}{P_g(x)}\, dx = \mathbb{E}_{x\sim P_r}[log\, \frac{P_r(x)}{P_g(x)}]$$

(KLは非対称)

$P_r(x) > 0\ and\ P_g(x) \to 0$

- 真のデータを生成する確率が低い
- Mode dropping
- $KL \to \infty$

$P_g(x) > 0\ and\ P_r(x) \to 0$

- 真のデータにないようなデータを生成
- Fake looking
- $KL \to 0$

Jensen-Shannon Divergence

$$JSD(P_r||P_g) = \frac{1}{2} KL(P_r||P_A) + \frac{1}{2} KL(P_g||P_A) \quad , (P_A = \frac{P_g + P_r}{2})$$

- KLが非対称で不便なので対称にしたもの
- GANは実質これを最小化することで真のデータの分布を学習している

GAN

Original Loss

$$L(D, g_\theta) = \mathbb{E}_{x\sim P_r}[\log D(x)] + \mathbb{E}_{x\sim P_g}[\log(1 - D(x))] \qquad (1)$$

Dは式(1)を最大化、Gは式(1)を最小化

明らかに、$D^*(x) = \frac{P_r(x)}{P_r(x)+P_g(x)}$ のとき最大

$$\rightarrow L(D^*, g_\theta) = 2\,JSD(P_r||P_g) - 2\log 2 \qquad (2)$$

すなわち

- DはJSDを近似するよう学習
- GはDによって近似されたJSDを最小化するように学習

GANの問題

理想的には、

1. Dを精一杯学習させて精度よくJSDを近似する
2. Gを学習させる

を繰り返すのが良いのでは？

→ うまくいかない

- log(D(x))にするという小手先のトリックもあるが、それでも不安定

Question

- なぜDiscriminatorを更新すればするほどGeneratorの更新がうまくいかなくなるのか
- なぜGeneratorの更新式の- log(D(x))トリックで多少うまくいくのか
  - 実質何を最適化しているのか (JSDと似ているのか?)
- なぜGANの学習はそんなに不安定なのか
- どうにか回避できないのか

Sources of Instability

そもそも、Gを固定してDを学習させた場合

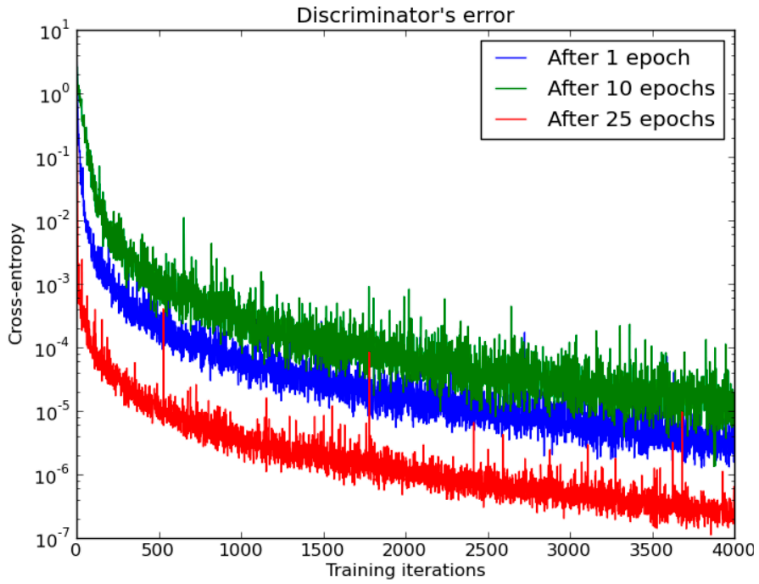
$$\min -L(D, g_\theta) = \min -\mathbb{E}_{x\sim P_r}[\log D(x)] + \mathbb{E}_{x\sim P_g}[\log(1 - D(x))]$$

$$= -2\,JSD(P_r \| P_g) + 2\log 2 \quad \text{であってほしい}$$

→ だが実際にはどこまでも小さくなり最終的には 0 に

→ なぜ？？可能性としては、、

- 分布が不連続
- 互いに素な台（support)が存在する



Sources of Instability

- 真のデータ分布 P_r は低次元の多様体空間上に存在する
  - Ex. 256 × 256の画像だったら、256*256次元空間すべてに配置されるというよりは低次元な空間に集中しているはず

- 生成分布P_gも同様
  - 実際GANではランダムに生成された $z \sim P(z)$になんらかの関数を適用したもの $g(Z)$
  - 多くのケースではZの次元数 $(\cong 100)$はサンプルXのそれよりはるかに小さい
  - すなわち低次元多様体空間上に存在

Sources of Instability

optimal discriminatorが存在するための条件に関する証明が続きます
(全然追えませんでした)

最終的には、このどちらかが成り立てばよい

1. 二つの分布がdisjointなsupportを持つ場合
2. 二つの分布がそれぞれ低次元多様体に乗っている場合
   - 低次元多様体の共有空間は、測度0になるから (たぶん)

イメージ的には、例えば二次元空間では曲線は
基本的には点でしか交わらないので分離できるよって感じ
(一点の確率密度は0)

Sources of Instability

結局!

- 二つの分布が互いに素なsupportを持つ、もしくは
- 低次元多様体に存在している場合、

Optimal discriminatorが存在し、
そのとき $x \in X$に対し、

$$JSD = \log 2 \ , \quad \nabla_x D(x) = 0$$

要は自由にD決めてよかったら最終的には完璧に分離できちゃうぜってこと
つまりJSD自体に低次元多様体同士を近づける力はない

Sources of Instability

また、最適なD^*と学習途中のDがどれだけ離れているかをϵとおいて、

$$\left|\nabla_\theta \, E_{z\sim p(z)}\left[\log\left(1 - D\left(g_\theta(z)\right)\right)\right]\right|_2 \leq \mathrm{M}\frac{\epsilon}{1-\epsilon}$$

らしいです

結局!

D^* の近似精度が上がるほどgradientのnormは小さくなっていく

つまり、

- 学習が不足していると、正確でない JSDを最小化することに
- 学習しすぎると、Gradientがどんどん小さくなってしまう

→ それは難しいわけだ、という話

$-\log(D)$トリックについて

- Gのlossに$\log(1-D(x))$ ではなく$-\log(D(x))$を使うトリック

- 学習初期は超簡単にDiscriminatorによって分離できてしまうので、
  さっき述べた問題から、log(1-D)の勾配がどんどん小さくなる
  （このトリック使わないと学習進まないらしい）

- 式的にも、D(x)がほぼ0だったとき、
  $\nabla\log(1-D(x)) = -\frac{\nabla D(x)}{(1-D(x))}$ よりも $-\nabla\log(D(x)) = -\frac{\nabla D(x)}{D(x)}$ のほうが大きそう

- これって何を最小化してることになるの？

$-\log(D)$トリックについて

実は、、

$$\mathbb{E}_{z\sim p(z)}\left[-\nabla_\theta \log D^*\left(g_\theta(z)\right)|_{\theta=\theta_0}\right]$$
$$= \nabla_\theta \left[\, KL\left(P_{g_\theta}||P_r\right) - 2\, JSD\left(P_{g_\theta}||P_r\right)\big|_{\theta=\theta_0}\right]$$

- JSDの符号直感と逆じゃん。。
- もう一つはMaximum Likelihoodのときと逆のKL!!
  - Fake looking -> high cost
  - Mode drop -> low cost

→ クオリティは良くなる方向に動くが、mode dropに対してlossが働かない

→ practiceに合うね！

また計算していくと、この勾配の分散は学習が進むに連れて発散することがわかった

→ 勾配法が不安定に！

Kullback-Leibler (KL) divergence 復習

$$KL(P_r||P_g) = \int_x P_r(x)\, log \frac{P_r(x)}{P_g(x)}\, dx = \mathbb{E}_{x\sim P_r}[log \frac{P_r(x)}{P_g(x)}]$$

(KLは非可換)

$P_r(x) > 0\ and\ P_g(x) \to 0$

- 真のデータを生成する確率が低い
- Mode dropping
- $KL \to \infty$

$P_g(x) > 0\ and\ P_r(x) \to 0$

- 真のデータにないようなデータを生成
- Fake looking
- $KL \to 0$

Solution

- じゃあどうすればいいの？？

- 解決策 1
  - Dを学習させるとき、二つの分布にノイズを加える

- 解決策 2
  - JSDではなく他の分布間距離を測る指標を利用する

解決策 1

- Dを学習させるとき、二つの分布にノイズを加える → $D^*(x) = \frac{P_{r+\epsilon}(x)}{P_{r+\epsilon}(x)+P_{g+\epsilon}(x)}$
  - 色々証明がんばってました。謎でした。
  - そもそも問題として、VAEみたいに最終出力にノイズを仮定してないからsupportがoverlapしない
- ノイズ加えない場合は、二つの分布が割と似てても (Ex. きれいな画像を生成できていても) 完全に識別できるDが学習できてしまっていた
- ノイズをいい感じに加えれば、ほぼ完全に二つの分布がoverlapするので、Dを学習させればさせるほど良い

しかし、

→ ノイズの大きさを調整するの面倒

→ 実際は$P_{r+\epsilon}(x), P_{g+\epsilon}(x)$の距離じゃなくて$P_r(x), P_g(x)$ の距離を小さくしたい (ぼやけちゃう)

解決策 2

- JSDではなく他の分布間距離を測る指標を利用する
- ここでWasserstein Metrics (別名 Earth Mover Distance (EMD))

$$W(P,Q) = \inf_{\gamma \in \Gamma} \int_{x \times x} |x - y|_2 \, d\gamma(x,y)$$

  - 元々、最適輸送問題に使われる計量らしい。よくわからないです。
  - Pにある土を動かしてQにするコスト
  - Pのx地点の密度P(x)を$\gamma(x,y)$分だけy地点に移して最終的にy地点の密度がQ(y)になるように配分
  - $|x-y|$ がx地点からy地点に密度を移すコスト
  - 一番移動コストの低い配分γでのコスト　的な感じらしい

嬉しいのは、supportが違ってもコストが連続的に定義できること！

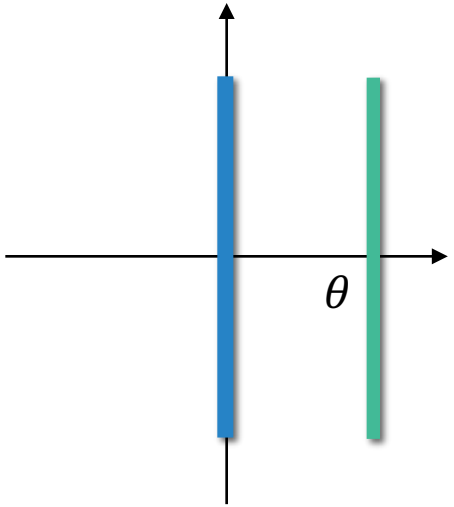
低次元多様体同士の距離をいい感じに測れる！

- 一つ目の論文はここで終わり
- 結論は特になかったです

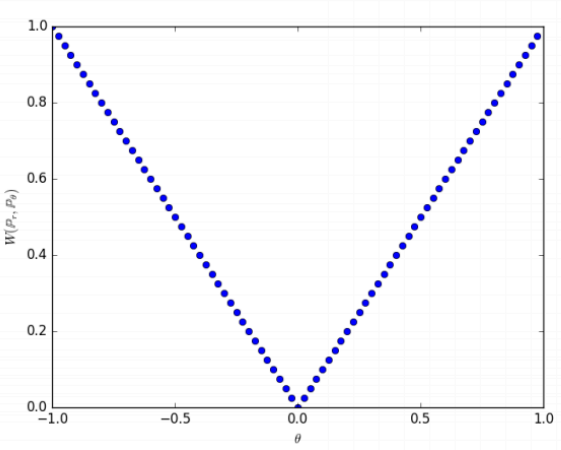
- ここからWasserstein GAN です

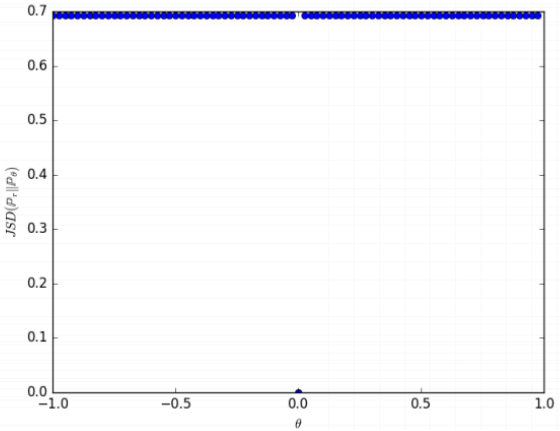
分布間距離比較

- 例
  - 2次元空間で、直線上の確率密度を持つ分布
    - P_r = (0, z) ($z \sim U(-1,1)$
    - $g_\theta(z) = (\theta, z)$
  - の分布間距離をEMD, KL, JSD, TVで計算



- $W(\mathbb{P}_0, \mathbb{P}_\theta) = |\theta|,$
- $JS(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- $KL(\mathbb{P}_\theta \| \mathbb{P}_0) = KL(\mathbb{P}_0 \| \mathbb{P}_\theta) = \begin{cases} +\infty & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ , \end{cases}$
- and $\delta(\mathbb{P}_0, \mathbb{P}_\theta) = \begin{cases} 1 & \text{if } \theta \neq 0\ , \\ 0 & \text{if } \theta = 0\ . \end{cases}$





WGAN

- 低次元多様体にのる分布間距離を測るには、
  JSDよりもEMDを利用するのが良いのでは？
- $W(P,Q) = \inf_{\gamma \in \Gamma} \int_{x \times x} |x - y|_2 \, d\gamma(x,y)$ ← これどうやって解くの？？
- Kantorovich-Rubinstein duality

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

- 1-リプシッツを満たす関数f_wを選んだなら、

$$W(P_r, P_\theta) = \max_{w \in W} \mathbb{E}_{x \sim P_r}[f_W(x)] - \mathbb{E}_{z \sim p_Z}[f_w(g_\theta(z))]$$

よく見るとGANとほぼ同じ！！！！(logがない)

WGAN

- リプシッツ連続性
  - 関数の傾きが有界に収まる

  - Feed-forward neural networkが 1-リプシッツを満たすためには？
    - Sigmoid, relu とかは大体リプシッツ連続
    - 各パラメータがコンパクト空間に収まっていれば良い

つまり

各パラメータのnormが

ある値c以下（論文中では0.01に設定)に収まっていれば良い

WGAN

結局

$$W(P_r, P_\theta) = \mathbb{E}_{x\sim P_r}[f_W(x)] - \mathbb{E}_{z\sim p_Z}[f_w(g_\theta(z))]$$

- Critic（元のDiscriminatorと区別するためにCriticと表現）
  - 収束するまで学習させる (論文ではCritic5回 G 1回)
  - 生出力を利用 (softplusとかlogとか何もいらない)
  - パラメータをc以下にclipping
- Generator
  - Criticと同じlossを利用する

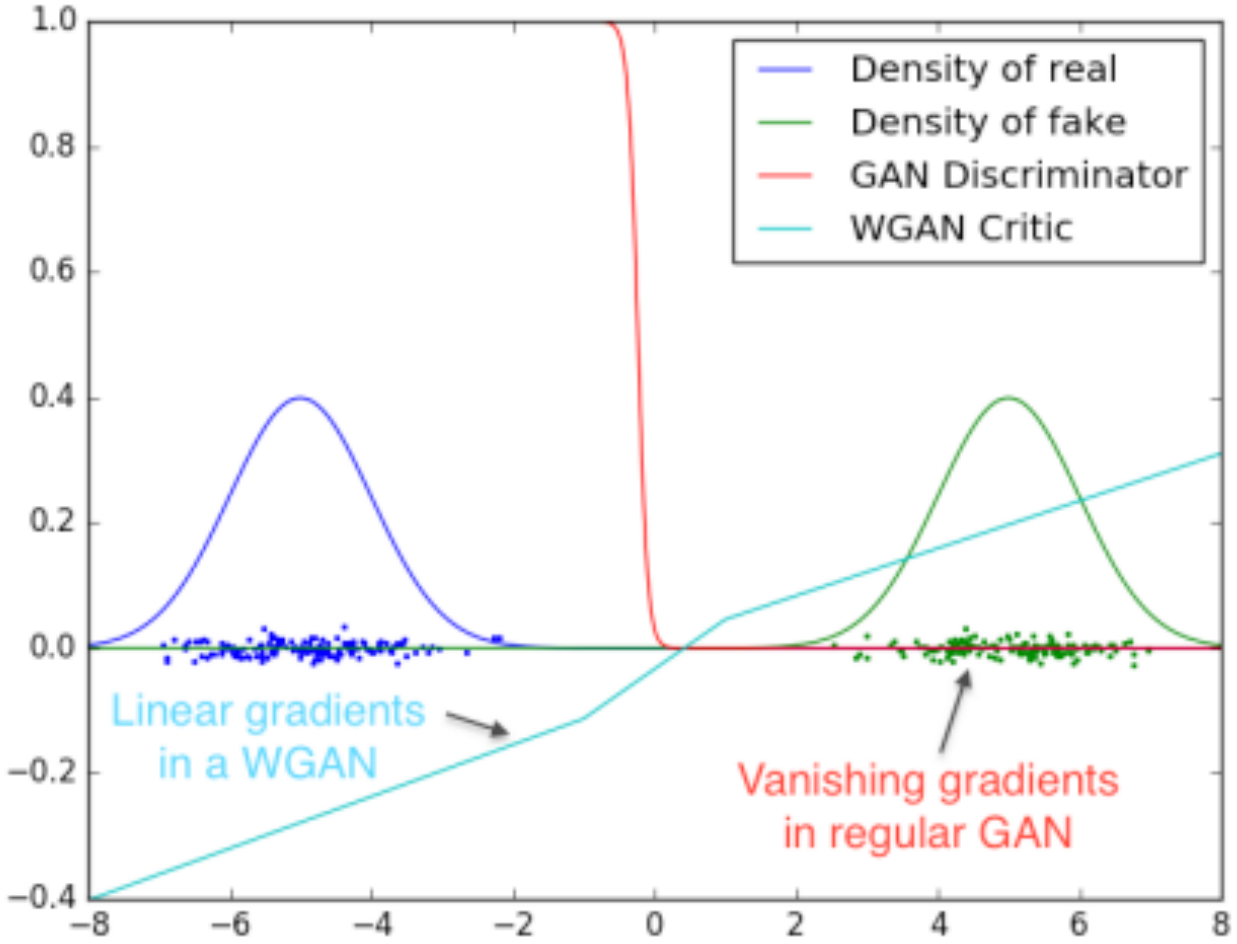
従来のGAN loss

- $L(f_w\, g_\theta) = \mathbb{E}_{x\sim P_r}[\log f_w(x)] + \mathbb{E}_{z\sim p_Z}[\log(1 - f_w(g_\theta(z)))]$
- $(\nabla_\theta\, L(f_w, g_\theta) = -\nabla_\theta \log f_w\left(g_\theta(z)\right))$

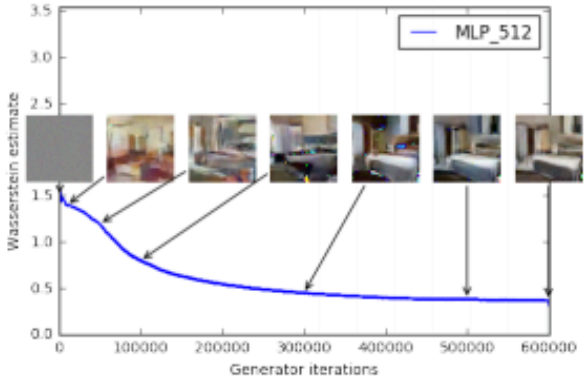
WGAN

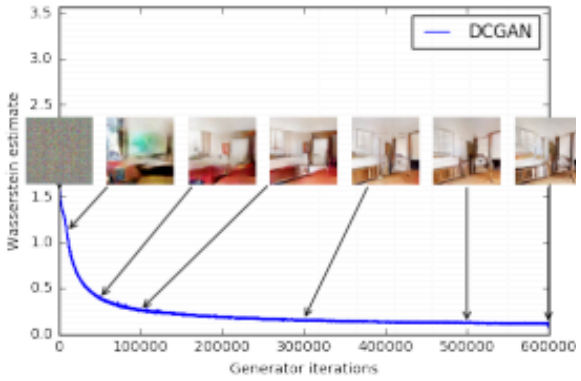
- 特徴
  - 学習が安定
    - 常に収束させるまで学習させても勾配が消えないのでDとGの学習スケジュールの調整が不要
    - BatchNormalizationがなくても動く。極端にはMLPでも学習可能
  - 意味のあるloss metrics
    - 生成クオリティの向上とlossの減少が相関する
    - 今までのGANはそもそもDとGで最適化するloss違ったうえに、各lossの値がめちゃくちゃ
    - このlossが小さければ良いモデルとして良いのでハイパラ探索の基準にできる (BOとか)
  - Mode Collapseは起こらない
    - これはそもそもfixされたDに対して最適なGを学習してしまうことによるもの
    - その時点でのGに対して収束するまでDを学習させるため大丈夫

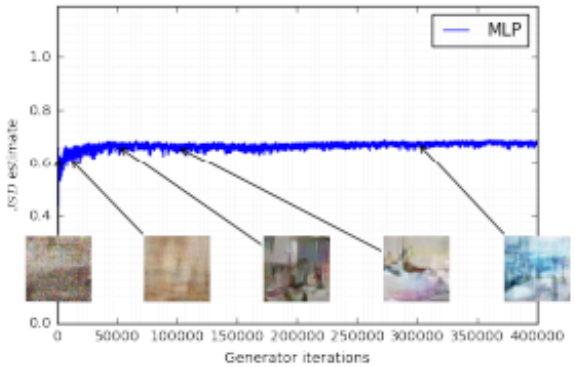
Don’t saturate

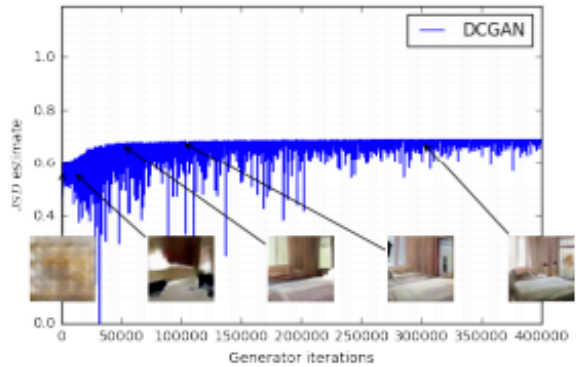


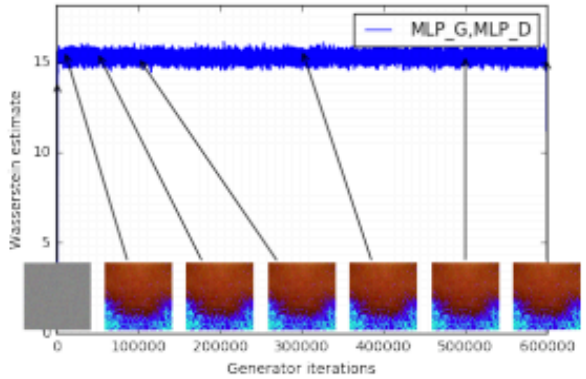
Meaningful Loss Metrics

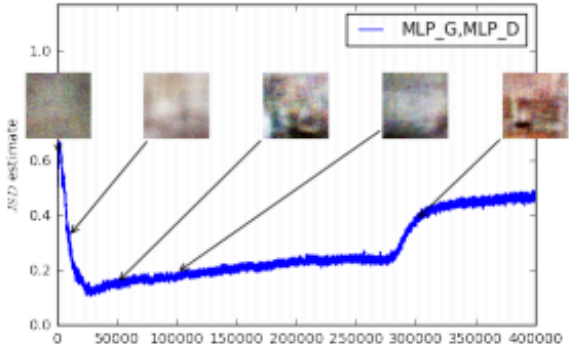












WGAN

- 実装上は、今までのに修正を加える程度
  - Dをclipping
  - Dの出力の非線形関数を外す
- 注意点
  - 学習率を高くしたり、初期化を間違えるとCriticが死ぬ（全パラメータ -0.01か0.01に)
  - Clippingしているためか、Motentum項を加えると学習が壊れる
    - 論文中ではRMSPropを推奨
  - 強めのWeight Decayでも良いらしい (結局抑えられればなんでもよい）

結論

- 著者頭良い & WGANすごい

- 一応実装してみたので興味あったら見てください
  - https://github.com/fukuta0614/chainer-image-generation/tree/master/WassersteinGAN
  - やってみた感じ、単純に[-0.01, 0.01]のclipだとだいぶモデルに制約加えてるので
    従来と同じ構造だとやや表現力不足する感
  - Clipping parameterをannealさせたり、層深くして構造を複雑させたりする必要がありそう