

RCP211 – Modèles génératifs

Réseaux génératifs antagonistes

Arnaud Breloy arnaud.breloy@lecnam.net

16 octobre 2024

Conservatoire national des arts & métiers

1. Principe

2. Optimisation des GAN

3. GAN conditionnels

Principe

Generative Adversarial Networks

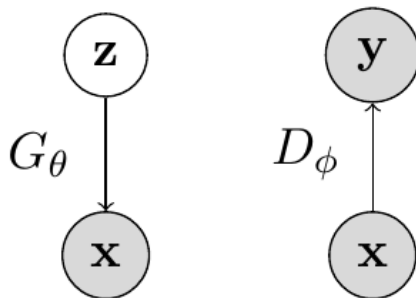
On parle de réseaux génératifs antagonistes pour décrire un type de modèle génératif profond introduits par Goodfellow et al. en 2014.

Les GAN possèdent deux composants :

- un générateur G de paramètres θ ,
- un discriminateur D de paramètres ϕ .

G et D sont des réseaux de neurones (fonctions paramétriques non-linéaires et différentiables).

Structure des GAN



- Générateur G_θ : génère de façon **déterministe** une observation x à partir d'un bruit z ,
- Discriminateur D_ϕ : distingue les exemples réels des exemples synthétiques.

Apprentissage du GAN

Le générateur et le discriminateur sont appris l'un contre l'autre et jouent à un jeu minimax à deux joueurs :

$$\min_{\theta} \max_{\phi} V(G_{\theta}, D_{\phi}) = \underbrace{\mathbb{E}_{\mathbf{x} \sim \mathbf{p}_{\text{data}}} [\log D_{\phi}(\mathbf{x})]}_{\text{score des données réelles}} + \underbrace{\mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\log (1 - D_{\phi}(G_{\theta}(\mathbf{z})))]}_{\text{score des données générées}}$$

Antagonisme

En décomposant, il y a donc deux tâches :

- $\max_{\phi} \mathbb{E}_{\mathbf{x} \sim \mathbf{p}_{\text{data}}} [\log D_{\phi}(\mathbf{x})] + \mathbb{E}_{\hat{\mathbf{x}} \sim p(\hat{\mathbf{x}})} [\log (1 - \hat{\mathbf{x}})]$: trouver les paramètres ϕ de D qui associent le score 1 aux données réelles et 0 aux données produites par G .
- $\min_{\theta} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\log (1 - D_{\phi}(G_{\theta}(\mathbf{z})))]$: trouver les paramètres θ de G qui produisent un $\hat{\mathbf{x}} = G_{\theta}(\mathbf{z})$ tel que D lui associe le score 1 ("réel")

Apprentissage du GAN

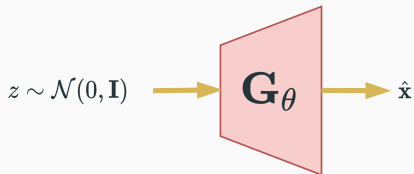
Le générateur et le discriminateur sont appris l'un contre l'autre et jouent à un jeu minimax à deux joueurs :

$$\min_{\theta} \max_{\phi} V(G_{\theta}, D_{\phi}) = \underbrace{\mathbb{E}_{\mathbf{x} \sim \mathbf{p}_{\text{data}}} [\log D_{\phi}(\mathbf{x})]}_{\text{score des données réelles}} + \underbrace{\mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\log (1 - D_{\phi}(G_{\theta}(\mathbf{z})))]}_{\text{score des données générées}}$$

Antagonisme

En décomposant, il y a donc deux tâches :

- $\max_{\phi} \mathbb{E}_{\mathbf{x} \sim \mathbf{p}_{\text{data}}} [\log D_{\phi}(\mathbf{x})] + \mathbb{E}_{\hat{\mathbf{x}} \sim p(\hat{\mathbf{x}})} [\log (1 - \hat{\mathbf{x}})]$: trouver les paramètres ϕ de D qui associent le score 1 aux données réelles et 0 aux données produites par G .
- $\min_{\theta} \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\log (1 - D_{\phi}(G_{\theta}(\mathbf{z})))]$: trouver les paramètres θ de G qui produisent un $\hat{\mathbf{x}} = G_{\theta}(\mathbf{z})$ tel que D lui associe le score 1 ("réel")



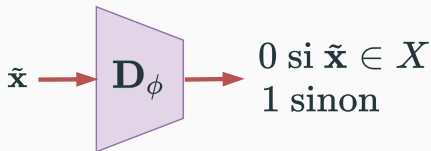
Génération

Le générateur transforme un bruit $\mathbf{z} \in \mathcal{N}(0, 1)^d$ en une observation synthétique $\hat{\mathbf{x}}$.

$\mathbf{z}$ est un code de l'espace latent $\mathcal{Z} = \mathbb{R}^d$.

L'analogie du faussaire

L'objectif de G est d'apprendre à tromper le discriminateur, c'est-à-dire à produire des observations $\hat{\mathbf{x}}$ telles que $p(\hat{\mathbf{x}}) = p(G(\mathbf{z}))$ est indiscriminable de $p(\mathbf{x})$.



Discrimination

Le discriminateur classe les observations x en deux catégories : réelle ou fausse. Sa sortie est une unique activation passée dans une sigmoïde :

$$f(x) = \frac{1}{1 + e^{-x}}$$

à valeurs dans $[0, 1]$.

Par convention, les données réelles ont un score de 1 et les données "fausses" ont un score de 0.

Apprentissage

Pour N epochs :

1. Échantillonner m de données réelles $\mathbf{x}_1, \dots, \mathbf{x}_m \sim \mathcal{D}$
2. Échantillonner m bruits $\mathbf{z}_1, \dots, \mathbf{z}_m \sim p_z$
3. Faire une étape du descente de gradient sur les paramètres θ du générateur ($\min_{\theta}$) :

$$\nabla_{\theta} V(G_{\theta}, D_{\phi}) = \frac{1}{m} \nabla_{\theta} \sum_{i=1}^m \log(1 - D_{\phi}(G_{\theta}(\mathbf{z}_i)))$$

4. Faire une étape de **montée** de gradient sur les paramètres ϕ du discriminateur ($\max_{\phi}$) :

$$\nabla_{\phi} V(G_{\theta}, D_{\phi}) = \frac{1}{m} \nabla_{\phi} \sum_{i=1}^m [\log D_{\phi}(\mathbf{x}_i) + \log(1 - D_{\phi}(G_{\theta}(\mathbf{z}_i)))]$$

Apprentissage

Pour N epochs :

1. Échantillonner m de données réelles $\mathbf{x}_1, \dots, \mathbf{x}_m \sim \mathcal{D}$
2. Échantillonner m bruits $\mathbf{z}_1, \dots, \mathbf{z}_m \sim p_{\mathbf{z}}$
3. Faire une étape du descente de gradient sur les paramètres θ du générateur ($\min_{\theta}$) :

$$\nabla_{\theta} V(G_{\theta}, D_{\phi}) = \frac{1}{m} \nabla_{\theta} \sum_{i=1}^m \log(1 - D_{\phi}(G_{\theta}(\mathbf{z}_i)))$$

4. Faire une étape de **montée** de gradient sur les paramètres ϕ du discriminateur ($\max_{\phi}$) :

$$\nabla_{\phi} V(G_{\theta}, D_{\phi}) = \frac{1}{m} \nabla_{\phi} \sum_{i=1}^m [\log D_{\phi}(\mathbf{x}_i) + \log(1 - D_{\phi}(G_{\theta}(\mathbf{z}_i)))]$$

Apprentissage

Pour N epochs :

1. Échantillonner m de données réelles $\mathbf{x}_1, \dots, \mathbf{x}_m \sim \mathcal{D}$
2. Échantillonner m bruits $\mathbf{z}_1, \dots, \mathbf{z}_m \sim p_{\mathbf{z}}$
3. Faire une étape du descente de gradient sur les paramètres θ du *générateur* ($\min_{\theta}$) :

$$\nabla_{\theta} V(G_{\theta}, D_{\phi}) = \frac{1}{m} \nabla_{\theta} \sum_{i=1}^m \log(1 - D_{\phi}(G_{\theta}(\mathbf{z}_i)))$$

4. Faire une étape de **montée** de gradient sur les paramètres ϕ du *discriminateur* ($\max_{\phi}$) :

$$\nabla_{\phi} V(G_{\theta}, D_{\phi}) = \frac{1}{m} \nabla_{\phi} \sum_{i=1}^m [\log D_{\phi}(\mathbf{x}_i) + \log(1 - D_{\phi}(G_{\theta}(\mathbf{z}_i)))]$$

Apprentissage

Pour N epochs :

1. Échantillonner m de données réelles $\mathbf{x}_1, \dots, \mathbf{x}_m \sim \mathcal{D}$
2. Échantillonner m bruits $\mathbf{z}_1, \dots, \mathbf{z}_m \sim p_{\mathbf{z}}$
3. Faire une étape du descente de gradient sur les paramètres θ du générateur ($\min_{\theta}$) :

$$\nabla_{\theta} V(G_{\theta}, D_{\phi}) = \frac{1}{m} \nabla_{\theta} \sum_{i=1}^m \log(1 - D_{\phi}(G_{\theta}(\mathbf{z}_i)))$$

4. Faire une étape de **montée** de gradient sur les paramètres ϕ du discriminateur ($\max_{\phi}$) :

$$\nabla_{\phi} V(G_{\theta}, D_{\phi}) = \frac{1}{m} \nabla_{\phi} \sum_{i=1}^m [\log D_{\phi}(\mathbf{x}_i) + \log(1 - D_{\phi}(G_{\theta}(\mathbf{z}_i)))]$$

Pourquoi le GAN fonctionne ? (1/2)

Équation du GAN :

$$\min_{\theta} \max_{\phi} V(G_{\theta}, D_{\phi}) = \mathbb{E}_{\mathbf{x} \sim \mathbf{p}_{\text{data}}} [\log D_{\phi}(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\log (1 - D_{\phi}(G_{\theta}(\mathbf{z})))]$$

Justification de la convergence

Pour un générateur G_{θ} fixé, le discriminateur D_{ϕ} est entraîné pour réaliser une classification binaire. On optimise :

$$\max_{\phi} \int_x (p_{\text{réelles}}(x) \log D(x) + p_{\text{fausses}}(x) \log(1 - D(x))) dx$$

qui admet pour maximum :

$$D_{\phi^*} : x \rightarrow \frac{p_{\text{réelles}}(x)}{p_{\text{réelles}}(x) + p_{\text{fausses}}(x)}$$

Pourquoi le GAN fonctionne ? (2/2)

Convergence des distributions

Pour G fixé $D^* = D_{\phi^*} : x \rightarrow \frac{p_{\text{réelles}}(x)}{p_{\text{réelles}}(x) + p_{\text{fausses}}(x)}$

La valeur optimale de V est donc :

$$\min_{\theta} V(G, D^*) = \int_x \left(p_r(x) \log \frac{p_r(x)}{p_r(x) + p_f(x)} + p_f(x) \log \frac{p_f(x)}{p_r(x) + p_f(x)} \right) dx$$

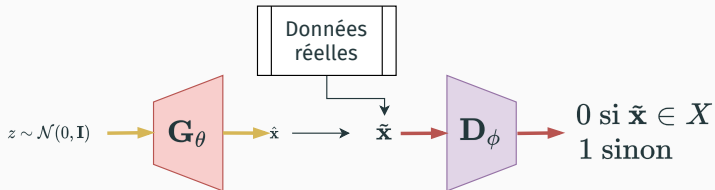
Et on peut montrer que :

$$\min_{\theta} V(G, D^*) = 2D_{JS}(p_r || p_f) - 2 \log 2$$

où D_{JS} est la divergence de Shannon-Jensen.

Optimiser le générateur revient donc à minimiser la divergence de Jensen-Shannon, c'est-à-dire à faire tendre vers $p_r = p_f$.

Schéma récapitulatif



Comparaison avec les autres modèles génératifs

Modèles explicites ou implicites

On considère généralement que les GAN sont des modèles génératifs **implicites**.

Contrairement aux modèles de mélange gaussiens ou aux auto-encodeurs variationnels, l'apprentissage ne se fait pas par une maximisation de la vraisemblance.

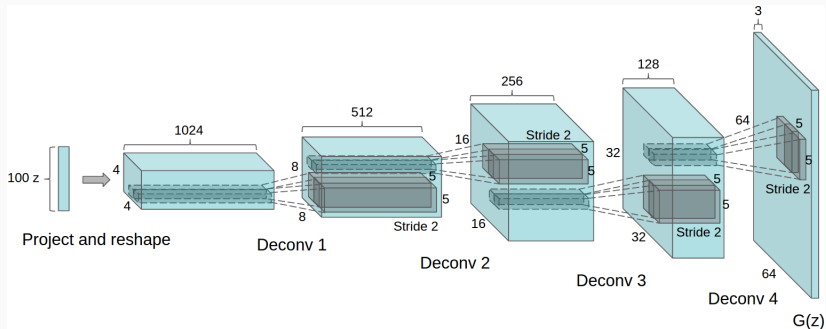
Le discriminateur comme mesure de distance

On utilise le discriminateur D comme test permettant de juger si les deux distributions (réelle et générée) sont statistiquement différentes.

→ D définit une mesure de distance *implicite* entre les deux distributions (qu'elle tente de maximiser tandis que G tente de la réduire).

DCGAN : *Deep Convolutional GAN*

DCGAN est une extension des GAN qui utilise une architecture convolutive pour le discriminateur et une architecture convolutive transposée pour le générateur.



Radford et al., 2015

Optimisation des GAN

Constat

L'optimisation minimax n'est généralement pas stable.

On observe en général trois types de problèmes :

- **Non-convergence** : les paramètres oscillent fortement sans se stabiliser.
- **Mode collapse** : le générateur produit des observations avec très peu de variété.
- **Rupture de l'équilibre** : le discriminateur devient « plus fort » que le générateur qui reçoit des gradients très faibles en retour.

Cette instabilité introduit une haute sensibilité des GAN au choix des hyperparamètres.

Constat

L'optimisation minimax n'est généralement pas stable.

On observe en général trois types de problèmes :

- **Non-convergence** : les paramètres oscillent fortement sans se stabiliser.
- **Mode collapse** : le générateur produit des observations avec très peu de variété.
- **Rupture de l'équilibre** : le discriminateur devient « plus fort » que le générateur qui reçoit des gradients très faibles en retour.

Cette instabilité introduit une haute sensibilité des GAN au choix des hyperparamètres.

Constat

L'optimisation minimax n'est généralement pas stable.

On observe en général trois types de problèmes :

- **Non-convergence** : les paramètres oscillent fortement sans se stabiliser.
- **Mode collapse** : le générateur produit des observations avec très peu de variété.
- **Rupture de l'équilibre** : le discriminateur devient « plus fort » que le générateur qui reçoit des gradients très faibles en retour.

Cette instabilité introduit une haute sensibilité des GAN au choix des hyperparamètres.

Équation du GAN

$$\min_{\theta} \max_{\phi} V(G_{\theta}, D_{\phi}) = \mathbb{E}_{\mathbf{x} \sim \mathbf{p}_{\text{data}}} [\log D_{\phi}(\mathbf{x})] + \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z})} [\log (1 - D_{\phi}(G_{\theta}(\mathbf{z})))]$$

→ jeu à somme nulle (*zero-sum game*)

→ solution : équilibre de Nash

L'équilibre de Nash n'est pas forcément atteignable par descente de gradient (surtout pour une fonction de coût non-convexe).

Les distributions réelles sont généralement **multimodales** (e.g. différents chiffres pour de MNIST).

Le *mode collapse* intervient quand le générateur ne produit plus que des échantillons d'un seul mode (par exemple, des images d'une seule classe).

Origine du *mode collapse*

Considérons le cas limite où l'on optimise uniquement G . Alors les observations générées convergent vers $\mathbf{x}^*$ qui trompe le mieux D . Dans le cas extrême :

$$\mathbf{x}^* = \arg \max_x D(x)$$

qui ne dépend pas de $\mathbf{z}$! La distribution $G(\mathbf{z})$ s'est effondrée en un seul point et le gradient est nul.

Fonction de coût du GAN

L'utilisation implicite de la divergence JS pour entraîner le générateur provoque des gradients évanescents. On rappelle que si le discriminateur est optimal, alors l'optimisation de G se fait sur :

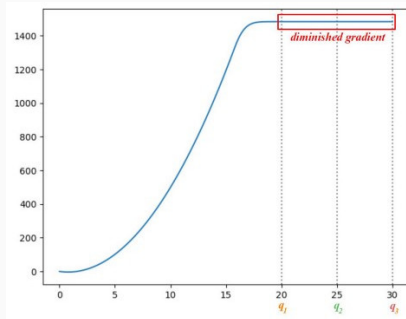
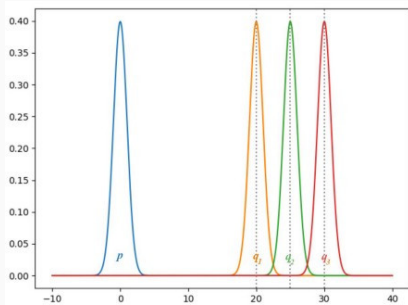
$$\min_G (G, D^*) = 2D_{JS}(p_r \| p_f) - 2 \log 2$$

Problème

Les gradients de la divergence JS diminuent très rapidement lorsque p_f est très proche ou très éloignée de p_r .

→ la convergence est lente, puis rapide, puis lente !

Illustration de l'évanescence des gradients de D_{JS}



À gauche : quatre distributions gaussiennes unidimensionnelles de moyennes différentes.

À droite : le valeur de $D_{JS}(p||q_i)$ lorsque la moyenne de q_i varie entre 0 et 30.

Optimisation du générateur

$$\nabla_{\theta} V(G) = \nabla_{\theta} \log(1 - D_{\phi}(G_{\theta}(z)))$$

Si $G_{\theta}(z)$ est peu plausible :

- $D_{\phi}(G_{\theta}(z)) \rightarrow 0,$
- $V(G) \simeq \log(1 - \varepsilon).$

L'erreur sature et les gradients deviennent évanescents.

Solution

On utilise :

$$\nabla_{\theta} - \log(D_{\phi}(G_{\theta}(z)))$$

Sur les données

- Normaliser les observations entre -1 et $+1$ (et utiliser $\tanh$ pour la dernière couche de G)
- Bruiter les données réelles et synthétiques ("augmentation" sur D)

Sur l'architecture

- Échantillonner z dans une gaussienne
- Interpoler sur les grands cercles plutôt que sur les lignes
- Éviter les gradients *sparse* : préférer *LeakyReLU* à *ReLU*, *average pooling* plutôt que *max pooling*, etc.

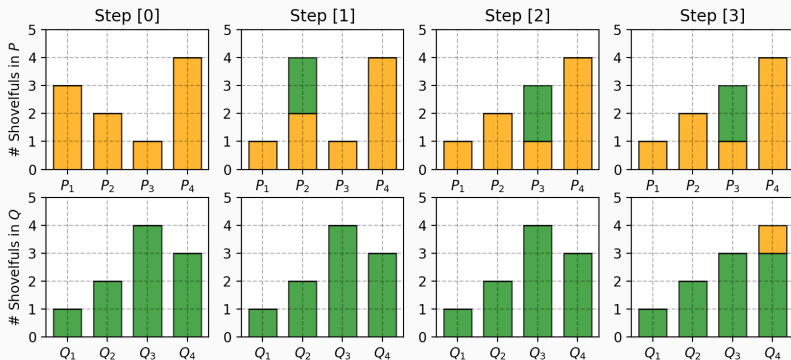
Sur l'optimiseur

- Utiliser Adam plutôt que SGD
- Ne pas mixer vraies et fausses données dans un *batch*
- *Label smoothing*

Distance de Wasserstein

L'utilisation de la divergence JS produit des gradients évanescents lorsque les distributions sont disjointes. Arjovsky et al. (2017) définit une nouvelle fonction de coût utilisant la **distance de Wasserstein**.

La distance de Wasserstein (ou *Earth Mover's distance*) correspond au transport optimal entre deux distributions :

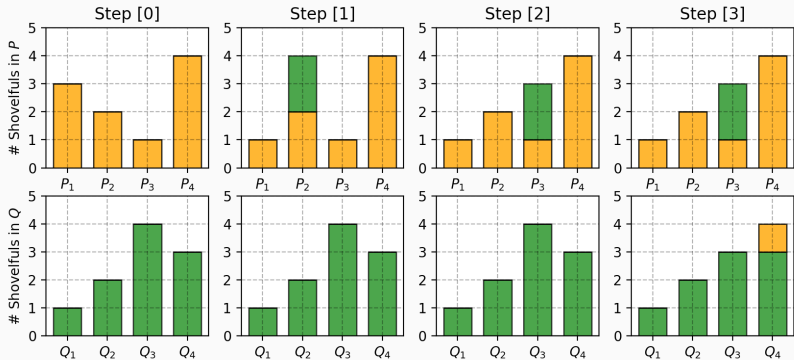


Distance de Wasserstein

1. déplace 2 unités de P_1 vers P_2
2. déplace 2 unités de P_2 vers P_3
3. déplace 1 unité de Q_3 vers Q_4

Distance de Wasserstein

nombre d'unités déplacées $\times$
distance de déplacement



Cas des probabilités continues

$$W(p_r, p_f) = \inf_{\gamma \sim \Pi(p_r, p_f)} \mathbb{E}_{(x, y) \sim \gamma} [\|x - y\|]$$

W croît linéairement avec la distance entre les moyennes des distributions, même si les distributions sont disjointes (ce qui n'est pas le cas de D_{JS}).

Inconvénient

On ne peut pas tester tous les probabilités conjointes $\Pi(p_r, p_f)$.

La dualité de Kantorovich-Rubinstein nous donne :

$$W(p_r, p_f) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim p_r} [f(x)] - \mathbb{E}_{x \sim p_f} [f(x)]$$

pour f l'ensemble des fonctions K -lipschitziennes (i.e.

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|).$$

Cas des probabilités continues

$$W(p_r, p_f) = \inf_{\gamma \sim \Pi(p_r, p_f)} \mathbb{E}_{(x,y) \sim \gamma} [\|x - y\|]$$

W croît linéairement avec la distance entre les moyennes des distributions, même si les distributions sont disjointes (ce qui n'est pas le cas de D_{JS}).

Inconvénient

On ne peut pas tester tous les probabilités conjointes $\Pi(p_r, p_f)$.

La dualité de Kantorovich-Rubinstein nous donne :

$$W(p_r, p_f) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim p_r} [f(x)] - \mathbb{E}_{x \sim p_f} [f(x)]$$

pour f l'ensemble des fonctions K -lipschitziennes (i.e.

$$|f(x_1) - f(x_2)| \leq K|x_1 - x_2|).$$

Implémentation du Wasserstein GAN

W-GAN en pratique

On impose que le discriminateur soit K -lipschitzien. Alors la fonction de coût devient :

$$L(p_r, p_f) = W(p_r, p_f) = \max_{\phi} \mathbb{E}_{x \sim p_r} [D_{\phi}(x)] - \mathbb{E}_{z \sim p(z)} [D_{\phi}(G_{\theta}(z))]$$

Autrement dit la fonction de coût revient à maximiser (pour le discriminateur) sur un *batch* m :

$$L(p_r, p_f) = \max_{\phi} \frac{1}{m} [D_{\phi}(x) - D_{\phi}(G_{\theta}(z))]$$

Comment rendre D K -Lipschitz ?

- restreindre les poids ϕ à une norme $[-c, c]$,
- utiliser la pénalité du gradient de Gulrajani et al., 2017

Implémentation du Wasserstein GAN

W-GAN en pratique

On impose que le discriminateur soit K -lipschitzien. Alors la fonction de coût devient :

$$L(p_r, p_f) = W(p_r, p_f) = \max_{\phi} \mathbb{E}_{x \sim p_r} [D_{\phi}(x)] - \mathbb{E}_{z \sim p(z)} [D_{\phi}(G_{\theta}(z))]$$

Autrement dit la fonction de coût revient à maximiser (pour le discriminateur) sur un *batch* m :

$$L(p_r, p_f) = \max_{\phi} \frac{1}{m} [D_{\phi}(x) - D_{\phi}(G_{\theta}(z))]$$

Comment rendre D K -Lipschitz ?

- restreindre les poids ϕ à une norme $[-c, c]$,
- utiliser la pénalité du gradient de Gulrajani et al., 2017

GAN conditionnels

Limitation du GAN standard

Le modèle de GAN vu précédemment reproduit une distribution $p(\mathbf{x})$.

Contrôle de la génération

Comment traiter le cas à n classes où l'on voudrait reproduire $p(\mathbf{x}|c)$?

- Option 1 : n modèles $\rightarrow$ lourd, pas forcément possible si peu d'exemples par classe.
- Option 2 : conditionner les distributions.

$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z}|\mathbf{y})))] .$$

GAN conditionnel

Deux changements principaux :

Générateur

On conditionne le générateur sur le vecteur de conditionnement c :

$$G(z, c) \rightarrow \hat{x}$$

Souvent on concatène simplement c au bruit z .

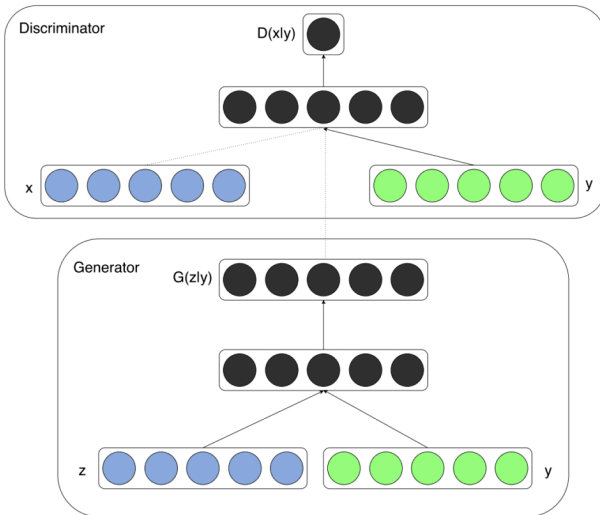
Discriminateur

On conditionne le discriminateur sur le vecteur de conditionnement c :

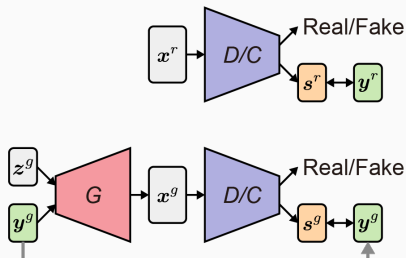
$$D(x, c) \rightarrow \text{score}$$

La plupart du temps, on introduit c dans une branche séparée.

Schéma du cGAN



AC-GAN : Auxiliary Classifier GAN



Cas d'application

Soit $\mathcal{D} = (\mathbf{X}, \mathbf{y})$ un jeu de données annoté. On ajoute une branche de sortie au discriminateur : un classifieur auxiliaire.

On définit deux fonctions de coût, la log-vraisemblance pour la source S et pour la classe C :

$$L_S = \mathbb{E}[\log \mathbb{P}(S = \text{réelle} | X_{\text{réelle}})] + \mathbb{E}[\log \mathbb{P}(S = \text{fausse} | X_{\text{fausse}})]$$

$$L_C = \mathbb{E}[\log \mathbb{P}(C = c | X_{\text{réelle}})] + \mathbb{E}[\log \mathbb{P}(C = c | X_{\text{fausse}})]$$

Le discriminateur maximise $L_S + L_C$

Le générateur maximise $L_C - L_S$.