GAN Foundations

CSC2541

Michael Chiu - chiu@cs.toronto.edu
Jonathan Lorraine - Lorraine@cs.toronto.edu
Ali Punjani - alipunjani@cs.toronto.edu
Michael Tao - mtao@dgp.toronto.edu

Basic Algorithm

Generative Models

Three major tasks, given a generative model *Q* from a class of models **Q**:

1. Sampling: drawing from *Q*
2. Estimation: find the *Q* in **Q** that best matches observed data
3. Evaluate Likelihood: compute *Q*(x) for a given x.

Generative Adversarial Networks: **specific** choice of **Q** (MLP) and **specific** choice of how to do estimation (adversarial).

Many other selections possible, and adversarial training is not limited to MLPs.

GANs can do (1) and (2) but not (3).

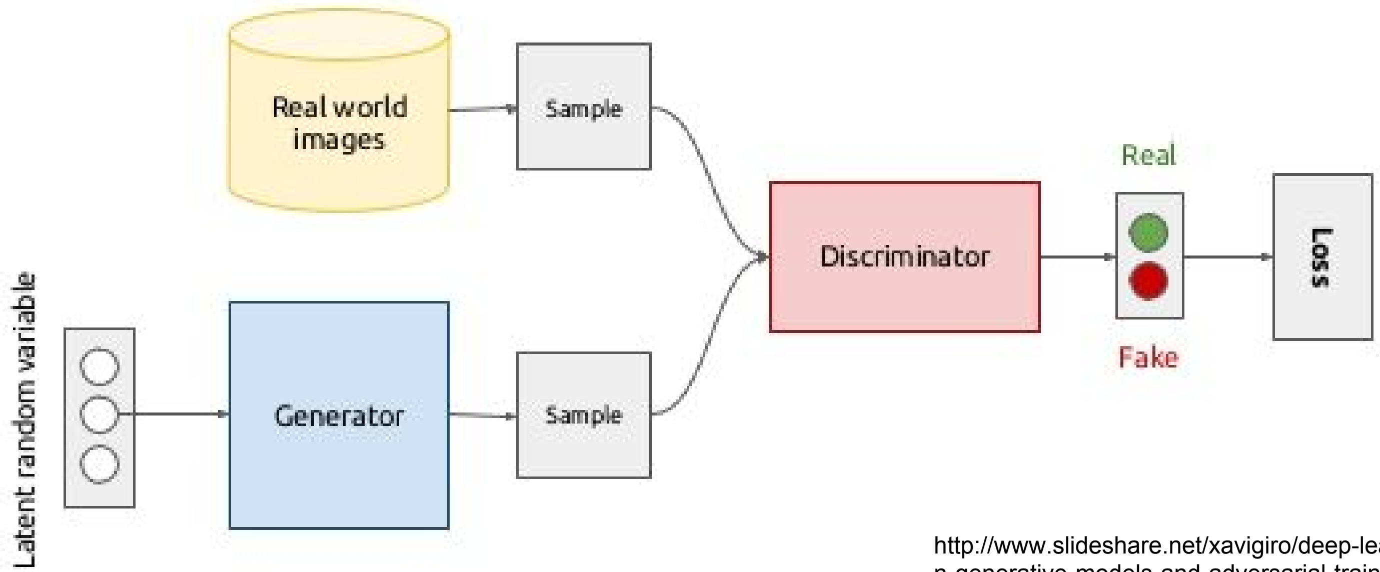
Big Idea - Analogy

- Generative: team of counterfeiters, trying to fool police with fake currency
- Discriminative: police, trying to detect the counterfeit currency

- Competition drives both to improve, until counterfeits are **indistinguishable** from genuine currency
- Now counterfeiters have as a side-effect learned something about real currency

Big Idea

- Train a generative model G(z) to generate data with random noise z as input
- Adversary is discriminator D(x) trained to distinguish generated and true data
- Represent both G(z) and D(x) by multilayer perceptrons for differentiability



http://www.slideshare.net/xavigiro/deep-learning-for-computer-visio
n-generative-models-and-adversarial-training-upc-2016

Formulation and Value Function

Latent variable z is randomly drawn from prior $p_z(z)$
Generator is a mapping from latent variable *z* to data space:

$G(z; \theta_g)$ Defined by MLP params θ_g

Discriminator is a scalar function of data space that outputs probability that input was genuine (i.e. drawn from true data distribution):

$D(x; \theta_d)$ Defined by MLP params θ_d

Trained with value function:

$$\min_G \max_D V(D, G) = \underbrace{\mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})]}_{\text{log prob of D predicting that real-world data is genuine}} + \underbrace{\mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]}_{\text{log prob of D predicting that G's generated data is not genuine}}.$$

Perspectives on GANs

1. *Want*: Automatic model checking and improvement

    Human building a generative model would iterate until the model generates plausible data. GAN ***attempts*** to automate that procedure.

2. “Adaptive” training signal

    Notion that optimization of discriminator will find and adaptively penalize the types of errors the generator is making

3. Minimizing divergence

    Training GAN is equivalent to minimizing Jensen-Shannon divergence between generator and data distributions. Other divergences possible too

Pros and Cons

Pros:

- Can utilize power of backprop
- No explicit intractable integral
- No MCMC needed
- Any (differentiable) computation (vs. Real NVP)

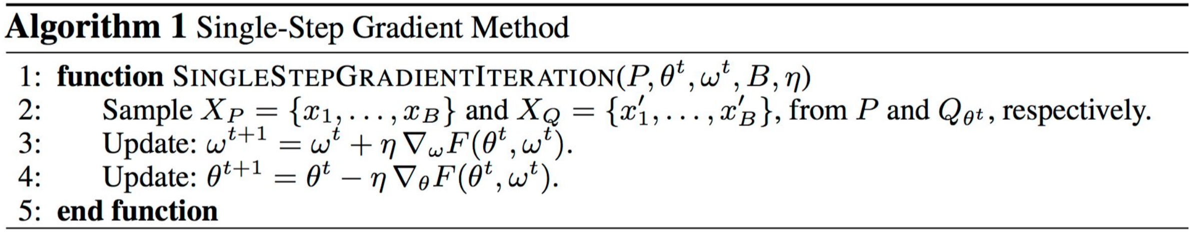
Pros and Cons

Cons:

- Unclear stopping criteria
- No explicit representation of $p_g(x)$
- Hard to train (immature tools for minimax optimization)
- Need to manually babysit during training
- No evaluation metric so hard to compare with other models (vs. VLB)
- Easy to get trapped in local optima that memorize training data
- Hard to invert generative model to get back latent z from generated x

Training a GAN

- Gibbs-type - like training procedure aka Block Coordinate Descent
    - Train discriminator (to convergence) with generator held constant
    - Train generator (a little) with discriminator held constant
- Standard use of mini-batch in practice
- Could train D & G simultaneously



Algorithm 1 Single-Step Gradient Method

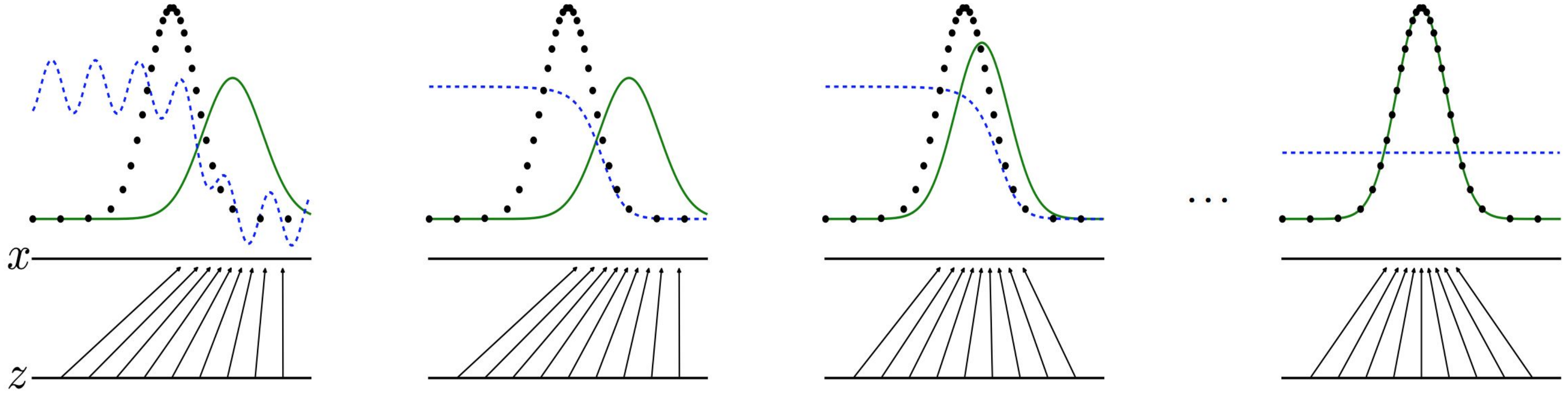
```
1: function SingleStepGradientIteration($P, \theta^t, \omega^t, B, \eta$)
2:     Sample $X_P = \{x_1, \ldots, x_B\}$ and $X_Q = \{x'_1, \ldots, x'_B\}$, from $P$ and $Q_{\theta^t}$, respectively.
3:     Update: $\omega^{t+1} = \omega^t + \eta \nabla_\omega F(\theta^t, \omega^t)$.
4:     Update: $\theta^{t+1} = \theta^t - \eta \nabla_\theta F(\theta^t, \omega^t)$.
5: end function
```

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

```
for number of training iterations do
    for $k$ steps do
        • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
        • Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
        • Update the discriminator by ascending its stochastic gradient:
            $$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$
    end for
    • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Update the generator by descending its stochastic gradient:
        $$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$
end for
```

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Alternating Training of D and G



GAN Convergence?

- How much should we train G before going back to D? If we train too much we won't converge (overfitting)
- Trick about changing the objective from min log(1-D(G(z))) to max log(D(G(z))) to avoid saturating gradients early on when G is terrible

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

log prob of D predicting that real-world data is genuine

log prob of D predicting that G's generated data is not genuine

Proof of optimality

- For a given generator, the optimal discriminator is:

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

Proof. The training criterion for the discriminator D, given any generator G, is to maximize the quantity $V(G, D)$

$$\begin{aligned} V(G,D) &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_z p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx \end{aligned} \tag{3}$$

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, the function $y \to a \log(y) + b \log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. The discriminator does not need to be defined outside of $Supp(p_{\text{data}}) \cup Supp(p_g)$, concluding the proof. $\square$

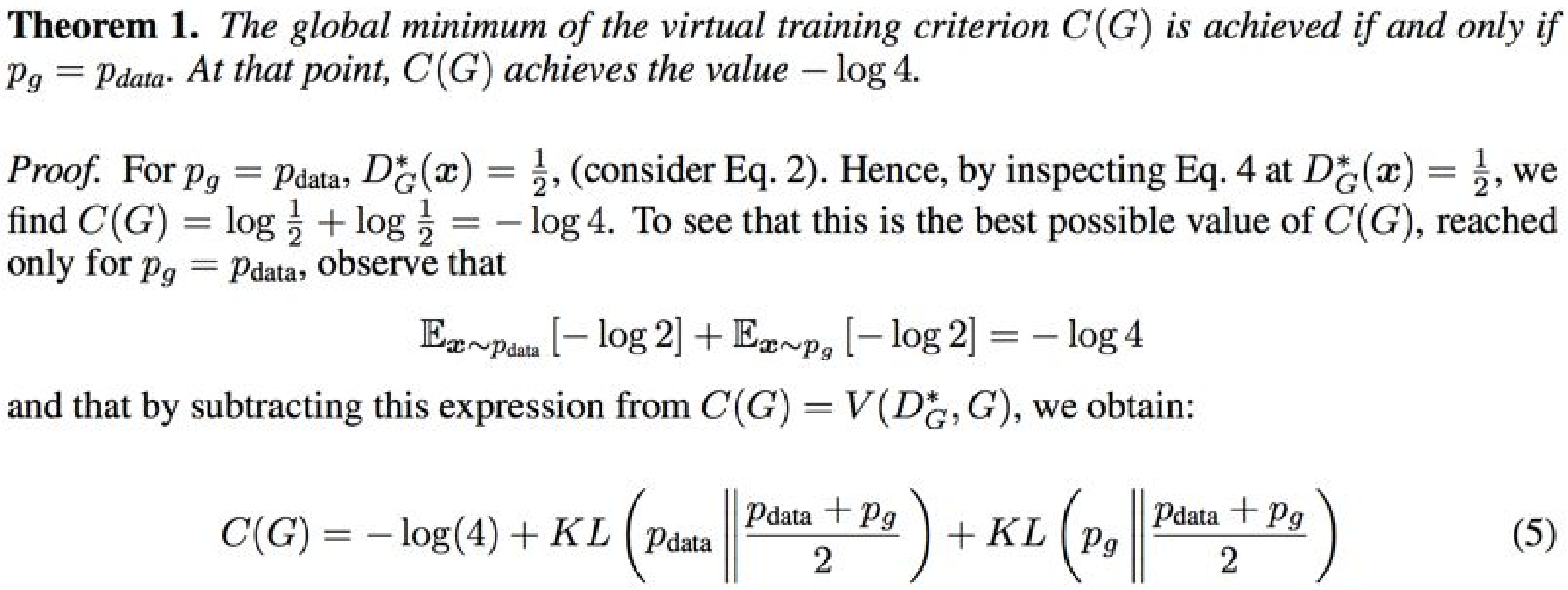
Proof of optimality

- Incorporating that into the minimax game to yield virtual training criterion:

$$
\begin{aligned}
C(G) =& \max_{D} V(G, D) \\
=& \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\
=& \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\
=& \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}
$$

Proof of optimality

- Equilibrium is reached when the Generator matches the data distribution



Theorem 1. *The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g = p_{data}$. At that point, $C(G)$ achieves the value $-\log 4$.*

Proof. For $p_g = p_{\text{data}}$, $D^*_G(\boldsymbol{x}) = \frac{1}{2}$, (consider Eq. 2). Hence, by inspecting Eq. 4 at $D^*_G(\boldsymbol{x}) = \frac{1}{2}$, we find $C(G) = \log \frac{1}{2} + \log \frac{1}{2} = -\log 4$. To see that this is the best possible value of $C(G)$, reached only for $p_g = p_{\text{data}}$, observe that

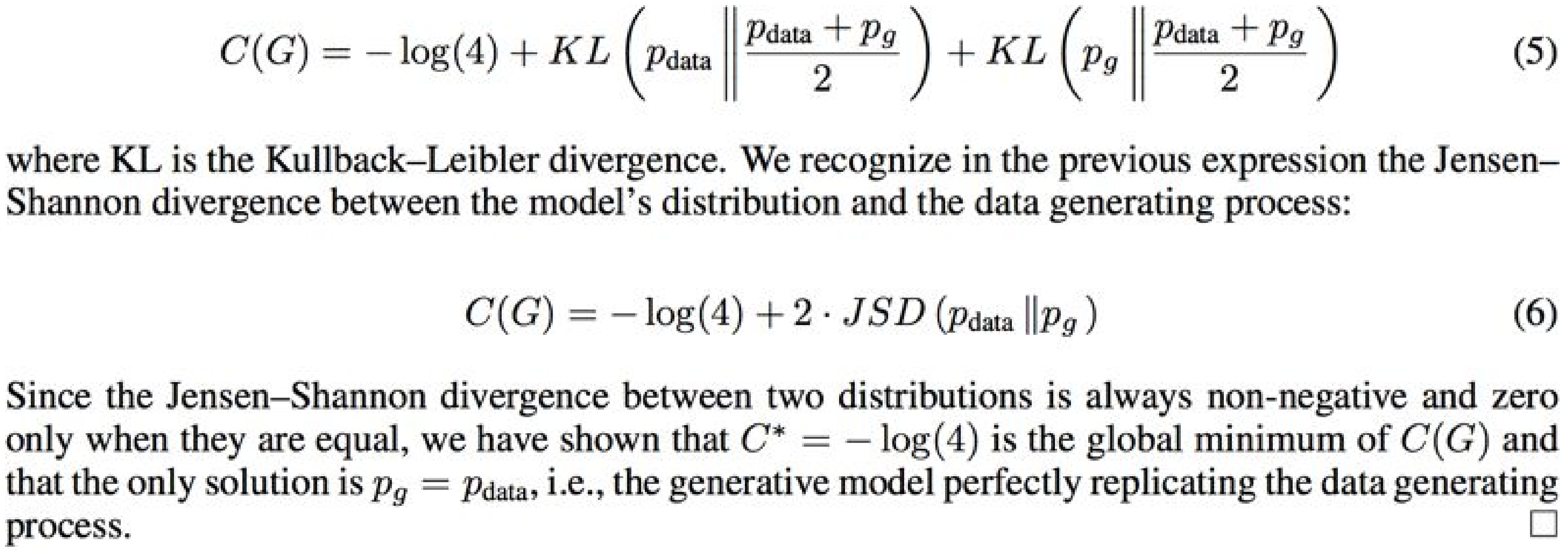
$$\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[-\log 2\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[-\log 2\right] = -\log 4$$

and that by subtracting this expression from $C(G) = V(D^*_G, G)$, we obtain:

$$C(G) = -\log(4) + KL\left(p_{\text{data}} \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right) + KL\left(p_g \left\| \frac{p_{\text{data}} + p_g}{2}\right.\right) \qquad (5)$$

Proof of optimality

- Virtual training criterion is JSD:



$$C(G) = -\log(4) + KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right) \tag{5}$$

where KL is the Kullback–Leibler divergence. We recognize in the previous expression the Jensen–Shannon divergence between the model's distribution and the data generating process:

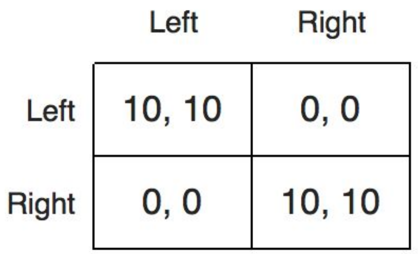
$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right) \tag{6}$$

Since the Jensen–Shannon divergence between two distributions is always non-negative and zero only when they are equal, we have shown that $C^* = -\log(4)$ is the global minimum of $C(G)$ and that the only solution is $p_g = p_{\text{data}}$, i.e., the generative model perfectly replicating the data generating process. $\square$

GANs as a NE

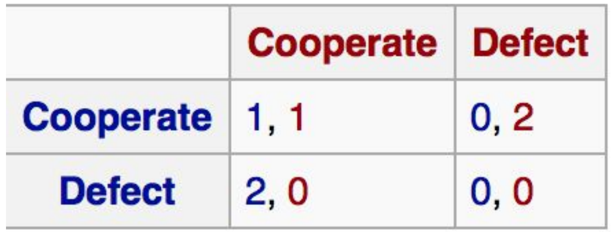
A game has 3 components - List of Players, potential actions by the players, and payoffs for the players in each outcome.

There are a variety of solution concepts for a game. A **Nash Equilibrium** is one, where each player does not want to change their actions, given the other players actions.

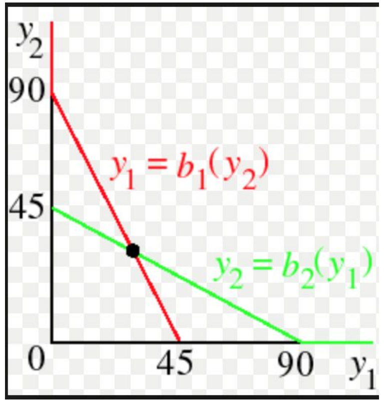


	Left	Right
Left	10, 10	0, 0
Right	0, 0	10, 10

***Fig. 2**: Choosing sides*



	Cooperate	Defect
Cooperate	1, 1	0, 2
Defect	2, 0	0, 0

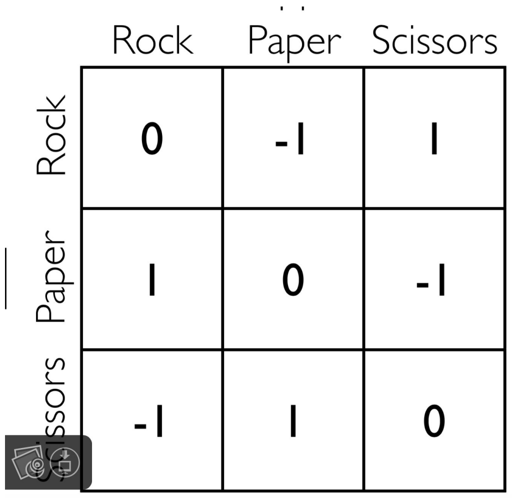


Mixed NE and Minimax

A game is minimax *iff* it has 2 players and in all states the reward of player 1 is the negative of reward of player 2.

$$\min_{x \in X} \max_{y \in Y} f(x, y) = \max_{y \in Y} \min_{x \in X} f(x, y).$$

Minimax Theorem states any point satisfying this is a PNE



	Rock	Paper	Scissors
Rock	0	-1	1
Paper	1	0	-1
Scissors	-1	1	0

If the opponent knows our strategy, it may be best to play a distribution of actions.

Can we construct a game, with a (mixed) equilibria that forces one player to learn the data distribution? Think counterfeiter vs police.

In the idealized game:

2 Players - Discriminator D and Generator G. Assume infinite capacity

Actions - G can declare a distribution in data space. D can declare a value (sometimes 0 to 1) for every point in data space.

Payoff - D wants to assign low values for points likely to be from G and high values for points likely to be from the real distribution. We could have payoff functions r_data(x) = log(D(x)) and r_g(x) = log(1 - D(x)):

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

In the **real** game:

2 Players - Discriminator D and Generator G. **Finite** capacity

Actions - G broadcasts m fake data points. D can declares a value for every fake and real (2m) point. Require both strategy sets to be differentiable, so use a neural network.

Payoff - Can only use approximations of expectation. “Similar” objective function for G?

- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)\right].$$

- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

train G to maximize $\log D(G(\boldsymbol{z}))$

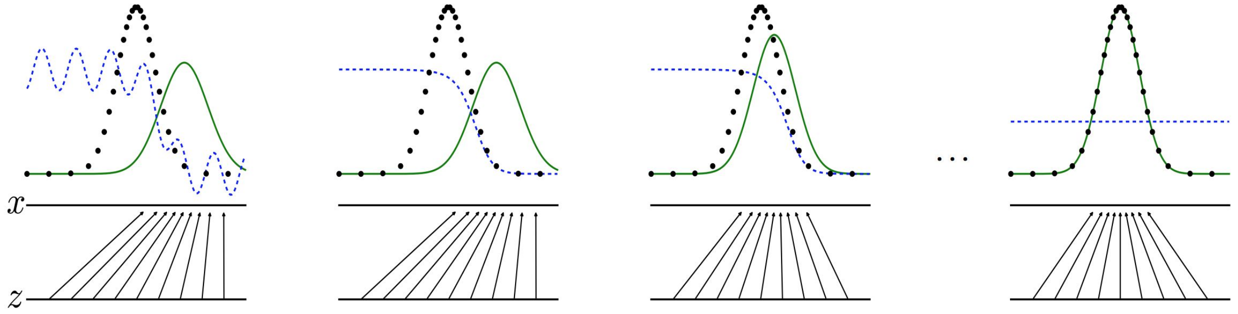
Unique PNE Existence for the idealized game.

If G plays some value more often than the Data, D will either (1) predict that point at a higher than average value, (2) predict the average, or (3) a below average value. In case (1) G will change its strategy by reducing mass in this region and moving it to a below average region. In case (2) and (3) D will increase its prediction of G in this region. Thus we are not at a PNE. A similar argument holds if G plays less often than Data. Thus p_G = p_Data at any PNE.

If D plays some value other than the average, then there exists some region above the average and some below. G will increase its payoff by decreasing its mass in low value region and moving it to the high value region. Thus D must play the same value at all points at a PNE (and that value expresses indifference between G and Data). D's payoff governs the value that expresses indifference and the loss that is learned (ex. p_r/(p_g+p_r) or p_g/p_r). If there is 1 value that expresses indifference the PNE is unique.

Existence?

Use Infinite capacity.



Global optimality

The PNE is a global min of the minimax equation.

One particular case is D(x) = p_r/(p_g+p_r) and G(x) maximizing JS(r || g), with payoff_r(D(x)) = log(D(x)) and payoff_g(D(x)) = log(1 - D(x)).

Another is is D(x) = p_g/p_r and G(x) maximizing KL(g || r), with payoff_r(D(x)) = D(x) - 1 and payoff_g(D(x)) = -log(D(x)).

Relations to VAE

- VAEs minimize an objective function indirectly. This is the ELBO.
- GANs attempt to minimize the objective function directly by training the discriminator to learn the objective function for a fixed Generator. How much can we change the generator, while still having the Discriminator as a good approximation?
- Framework for GANs can include alternative measures of divergences for the objective

Alternative Divergence Measures

So Far...

- We have the following based min-max problem using the objective

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

- When we have an optimal D with respect to G we obtain a Jenson-Shannon divergence term:

$$C(G) = \max_D V(G,D)$$

$$C(G) = -\log(4) + 2 \cdot JSD\left(p_{\text{data}} \| p_g\right)$$

However in implementation

- This formulation is difficult to train due to $\log(1 - D(G(z)))$aving poor convergence when the p_model differs from p_data too much
- Is replaced with -log $D(G(z))$ in the original paper.

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

Another alternative

- If we replace that term with

$$\log \frac{D(G(z;\theta_t);\psi_{t+1})}{1 - D(G(z;\theta_t);\psi_{t+1})}$$

- NOTATION: Rather than call G, we say x~Q for x=G(z), z~p_z
- NOTATION: Data is drawn from P
- We get a KL divergence term according to (recall that $D^*(x) = \frac{P(x)}{P(x)+Q(x)}$)

$$\mathrm{KL}[Q\|P] = \mathbb{E}_{x\sim Q} \log \frac{Q(x)}{P(x)}$$

$$= \mathbb{E}_{x\sim Q} \log \frac{1 - D^*(x)}{D^*(x)}$$

$$\approx \mathbb{E}_{x\sim Q} \log \frac{1 - D(x)}{D(x)}$$

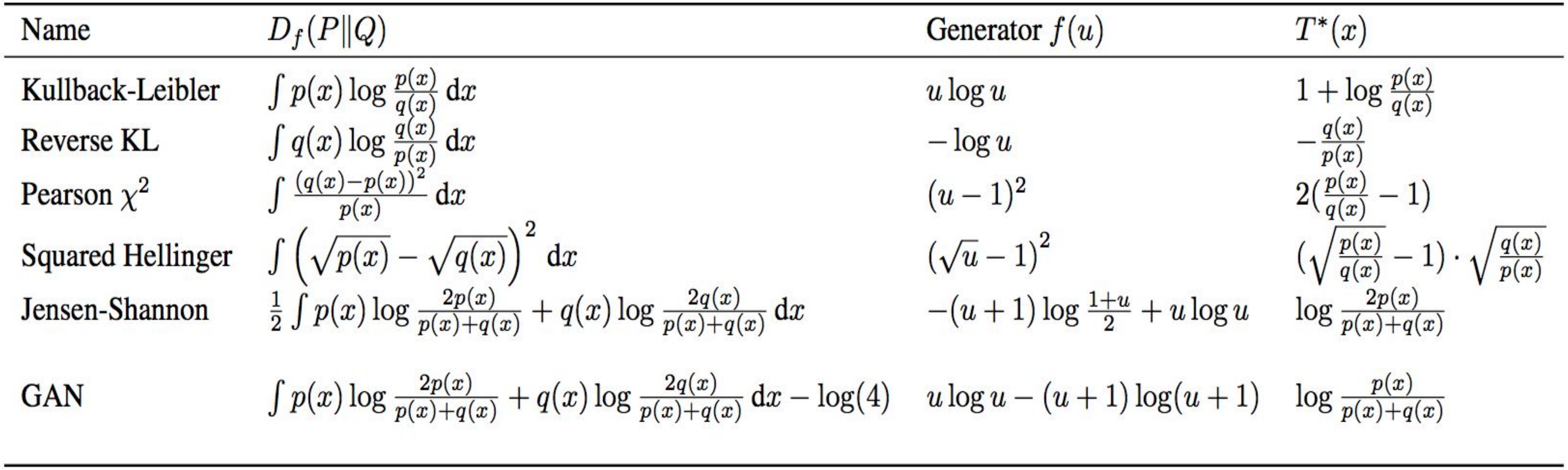
A family of alternatives (f-GAN)

- Consider a general class of divergences of the form

$$D_f(P\|Q) = \int_{\mathcal{X}} q(x) f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x,$$

- $f : \mathbb{R}_+ \to \mathbb{R}$ is a convex lower-semicontinuous such that f(1) = 0.
- Use *convex conjugates*, f^* to move from divergences to objectives
- Train a distribution Q and an approximation of divergence with T

Some divergence measures



Name	$D_f(P\|Q)$	Generator $f(u)$	$T^*(x)$
Kullback-Leibler	$\int p(x) \log \frac{p(x)}{q(x)} \,\mathrm{d}x$	$u \log u$	$1 + \log \frac{p(x)}{q(x)}$
Reverse KL	$\int q(x) \log \frac{q(x)}{p(x)} \,\mathrm{d}x$	$-\log u$	$-\frac{q(x)}{p(x)}$
Pearson χ^2	$\int \frac{(q(x)-p(x))^2}{p(x)} \,\mathrm{d}x$	$(u-1)^2$	$2(\frac{p(x)}{q(x)} - 1)$
Squared Hellinger	$\int \left(\sqrt{p(x)} - \sqrt{q(x)}\right)^2 \,\mathrm{d}x$	$(\sqrt{u} - 1)^2$	$(\sqrt{\frac{p(x)}{q(x)}} - 1) \cdot \sqrt{\frac{q(x)}{p(x)}}$
Jensen-Shannon	$\frac{1}{2} \int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \,\mathrm{d}x$	$-(u+1) \log \frac{1+u}{2} + u \log u$	$\log \frac{2p(x)}{p(x)+q(x)}$
GAN	$\int p(x) \log \frac{2p(x)}{p(x)+q(x)} + q(x) \log \frac{2q(x)}{p(x)+q(x)} \,\mathrm{d}x - \log(4)$	$u \log u - (u+1) \log(u+1)$	$\log \frac{p(x)}{p(x)+q(x)}$

Fenchel (Convex) Dual

$$f^*(t) = \sup_{u \in \mathrm{dom}_f} \{ut - f(u)\}.$$

Note that: $f = f^{**}$

The *f*-divergence is defined as: $$D_f(P\|Q) = \int_{\mathcal{X}} q(x) f\left(\frac{p(x)}{q(x)}\right) \mathrm{d}x,$$

Using the Fenchel Dual: $$\begin{aligned} D_f(P\|Q) &= \int_{\mathcal{X}} q(x) \sup_{t \in \mathrm{dom}_{f^*}} \left\{ t\frac{p(x)}{q(x)} - f^*(t) \right\} \mathrm{d}x \\ &\geq \sup_{T \in \mathcal{T}} \left(\int_{\mathcal{X}} p(x)\, T(x)\, \mathrm{d}x - \int_{\mathcal{X}} q(x)\, f^*(T(x))\, \mathrm{d}x \right) \\ &= \sup_{T \in \mathcal{T}} \left(\mathbb{E}_{x \sim P}\left[T(x)\right] - \mathbb{E}_{x \sim Q}\left[f^*(T(x))\right] \right), \end{aligned}$$

This poses divergence minimization into a min-max problem

New Optimization

- Now optimize T and Q with parameters ω and θ respectively:

$$F(\theta, \omega) = \mathbb{E}_{x\sim P}\left[T_\omega(x)\right] - \mathbb{E}_{x\sim Q_\theta}\left[f^*(T_\omega(x))\right]$$

$$T_\omega(x) = g_f(V_\omega(x))$$

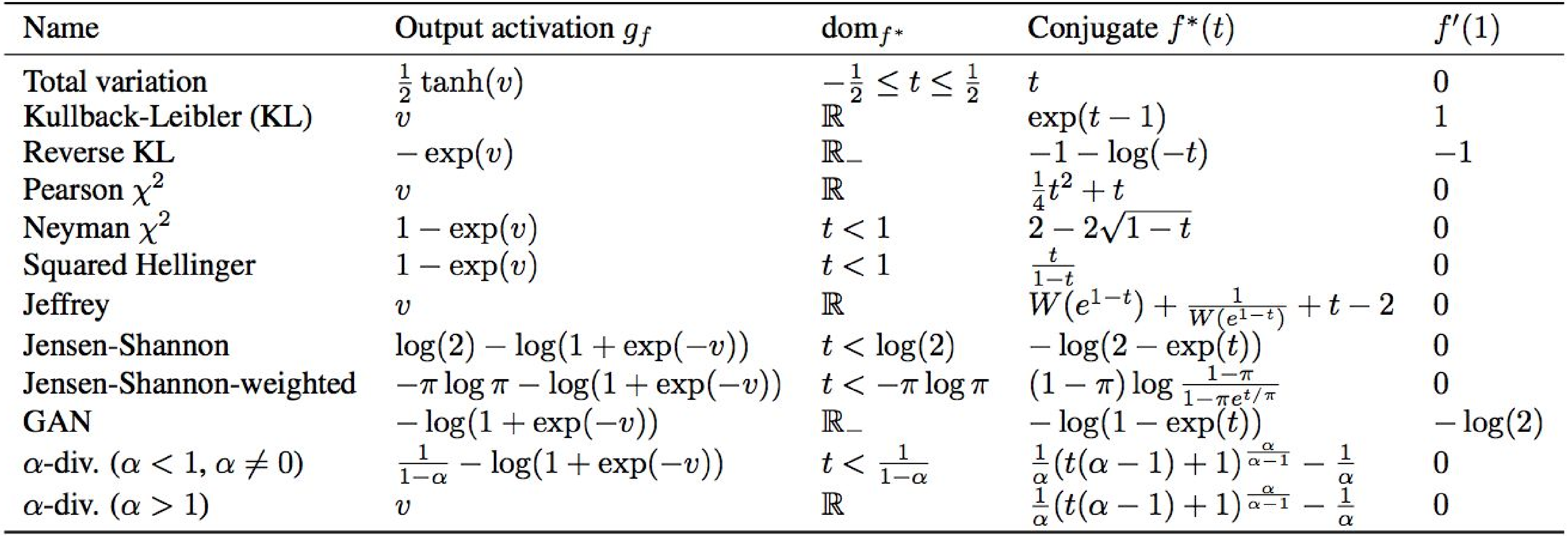
- g is a f-specific activation function

$$F(\theta, \omega) = \mathbb{E}_{x\sim P}\left[g_f(V_\omega(x))\right] + \mathbb{E}_{x\sim Q_\theta}\left[-f^*(g_f(V_\omega(x)))\right]$$

- For standard GAN: $g_f(v) = -\log(1 + e^{-v})$
  - With $D_\omega(x) \;=\; 1/(1 + e^{-V_\omega(x)})$

$$F(\theta, \omega) = \mathbb{E}_{x\sim P}\left[\log D_\omega(x)\right] + \mathbb{E}_{x\sim Q_\theta}\left[\log(1 - D_\omega(x))\right]$$

Fenchel Duals for various divergence functions



Name	Output activation g_f	dom_{f^*}	Conjugate $f^*(t)$	$f'(1)$
Total variation	$\frac{1}{2}\tanh(v)$	$-\frac{1}{2} \leq t \leq \frac{1}{2}$	t	0
Kullback-Leibler (KL)	v	$\mathbb{R}$	$\exp(t-1)$	1
Reverse KL	$-\exp(v)$	$\mathbb{R}_-$	$-1-\log(-t)$	-1
Pearson χ^2	v	$\mathbb{R}$	$\frac{1}{4}t^2+t$	0
Neyman χ^2	$1-\exp(v)$	$t<1$	$2-2\sqrt{1-t}$	0
Squared Hellinger	$1-\exp(v)$	$t<1$	$\frac{t}{1-t}$	0
Jeffrey	v	$\mathbb{R}$	$W(e^{1-t})+\frac{1}{W(e^{1-t})}+t-2$	0
Jensen-Shannon	$\log(2)-\log(1+\exp(-v))$	$t<\log(2)$	$-\log(2-\exp(t))$	0
Jensen-Shannon-weighted	$-\pi\log\pi-\log(1+\exp(-v))$	$t<-\pi\log\pi$	$(1-\pi)\log\frac{1-\pi}{1-\pi e^{t/\pi}}$	0
GAN	$-\log(1+\exp(-v))$	$\mathbb{R}_-$	$-\log(1-\exp(t))$	$-\log(2)$
α-div. ($\alpha<1, \alpha\neq 0$)	$\frac{1}{1-\alpha}-\log(1+\exp(-v))$	$t<\frac{1}{1-\alpha}$	$\frac{1}{\alpha}(t(\alpha-1)+1)^{\frac{\alpha}{\alpha-1}}-\frac{1}{\alpha}$	0
α-div. ($\alpha>1$)	v	$\mathbb{R}$	$\frac{1}{\alpha}(t(\alpha-1)+1)^{\frac{\alpha}{\alpha-1}}-\frac{1}{\alpha}$	0

For optimal T*, T* = f’(1)

f-GAN Summary

- **GAN can be generalized to minimize a large family of divergences (f-divergences)**
- The min-max comes from weakening the evaluation of D(P||Q) using the Fenshel dual
- Rather than as an adversarial network G/N, can see GAN as a system for simultaneously approximating the divergence (T) and minimizing the divergence (Q)