Boosting Generative Adversarial Networks

Haiyan Wang

Generative Adversarial Networks

Goodfellow et al, 2014

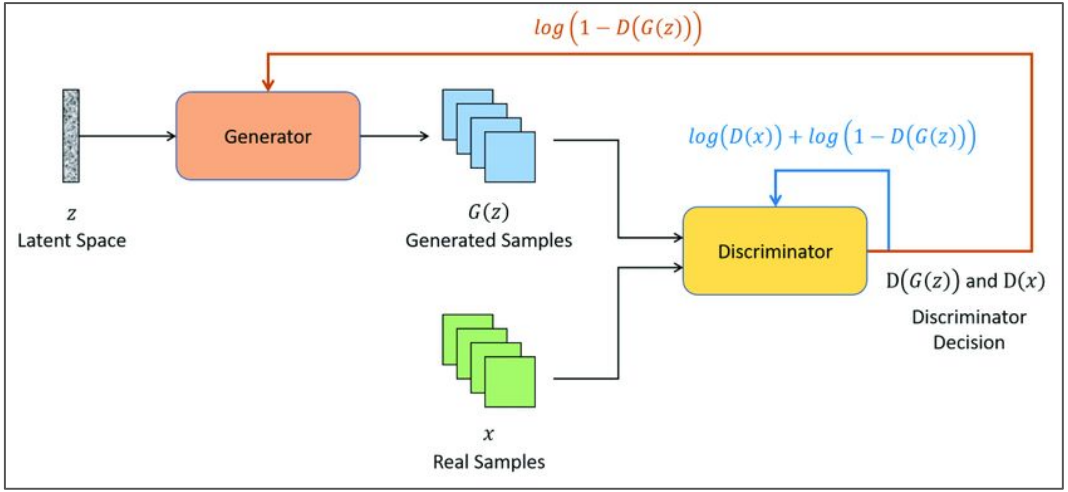
- Learning game: two-player minimax problem
  - Generator learns distribution and creates artificial samples from that distribution
  - Discriminator attempts to distinguish artificial samples from true data
  - Generator maximizes probability that discriminator misclassifies artificial samples as true data

- Consider a GAN implementing generator and discriminator as multilayer perceptrons
  - Draw input noise vectors $z \sim p_Z$
  - The generator is trained on real samples $x \sim p_X$ and returns artificial samples $G(z, \theta_G) \sim p_G$ where $G\colon p_Z \rightarrow p_G$ is a differentiable function approximated by the generator's MLP with parameters θ_z
  - Given unlabeled samples s, the discriminator returns scalars $D(s, \theta_D)$ representing the probability that $s \sim p_X$

$$V(D, G) = \mathbb{E}_{x \sim p_x}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

Generative Adversarial Networks (cont.)

- Train D to maximize probability of correctly categorizing both training examples and samples from G
  - D is attempting to maximize $V(D, G)$
- Train G to minimize probability that D correctly recognizes artificial samples
  - G is attempting to maximize $V(D, G)$, in particular $1 - D(G(z))$, the probability predicted by the discriminator that $G(z)$ is artificial



$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_x}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

Missing Modes Problem

- GANs often suffer from mode collapse in which they fail to generate samples from all but very specific regions of space
    - The generator is trained with the objective of producing "believable" outputs
    - If the generator produces an output that tricks the discriminator repeatedly, the discriminator should learn to always reject that output (optimal strategy)
    - If the discriminator becomes trapped in some local minimum in its loss function, it may not find this strategy

- The generator becomes over-optimized to a specific iteration of the discriminator rather than the entire system converging to a Nash equilibrium
- The resultant outputs are limited to certain classes

Strength of Weak Learnability

Kearns, 1988

- Hypothesis Boosting Problem - Does the existence of a "weak" classifier, or a model that performs slightly better than random guessing with high probability, implies the existence of a "strong" classifier, or a model whose accuracy can be arbitrarily high?

Schapire, 1990

- Strength of Weak Learnability - Any concept class is weakly learnable, i.e. there exists a hypothesis with slightly better than random performance, if and only if it is strongly learnable, i.e. there exists a hypothesis with arbitrarily low error

- Given some labeled binary dataset $X = \{(x_i, y_i)\}_{i=1}^n$ and some efficient weak learning algorithm A that produces weak classifiers $h : \mathcal{X} \to \{-1, 1\}$ where $x \subset \mathcal{X}$, we want to produce a classifier $H : \mathcal{X} \to \{-1, 1\}$ with error bounded above by some $\epsilon > 0$.

AdaBoost

[Freund & Schapire, 1995]

Initialize the weights on the training samples as a uniform distribution $d_{1,i} = \frac{1}{n}$

for classifier t of T **do**

- Generate weak classifier $h_t = A(X, d_t)$ on samples weighted by d_t
- Calculate the weighted misclassification error of h_t

$$\epsilon_t = \frac{1}{n} \sum_{i=1}^{n} d_{t,i} \mathbf{1}_{h_t(x_i) \neq y_i}$$

- Calculate the weight of h_t in the final strong classifier

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

AdaBoost (cont.)

Freund & Schapire, 1995

- Define a normalization factor Z_t such that updated weights are a discrete probability distribution

$$Z_t = \sum_{i=1}^{n} e^{\alpha_t y_i h_t(x_i)}$$

- Update sample weights where $y_i h_t(x_i) = 1$ if the prediction is correct and -1 if not

$$d_{t+1,i} = \frac{d_{t,i}}{Z_t} e^{\alpha_t y_i h_t(x_i)}$$

end for

Aggregate the weak classifiers

$$H = \sum_{t=1}^{T} \alpha_t \cdot h_t$$

AdaGAN

Tolstikhin et al, 2017

- Iteratively build a mixture of GANs with each new generator adjusting for mode collapse in previous iterations (see algorithm)
- Procedure outputs additive mixture of T generator functions with associated weights
  - Sample from mixture by sampling index from multinomial distribution (weights are probabilities) then sample from that generator

- Meta-algorithmic - applicable to many forms of generators, not just GANs

Input: Training sample $S_N := \{X_1, \dots, X_N\}$.
Output: Mixture generative model $G = G_T$.

Train vanilla GAN:
$W_1 = (1/N, \dots, 1/N)$
$G_1 = \mathrm{GAN}(S_N, W_t)$
for $t = 2, \dots, T$ **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

AdaGAN - Important Concepts and Results

- Possible to create optimal mixture of generators incrementally by training new generators on reweighted data - convergence is exponential under “weak learnability” assumptions
- GANs search for generators $G: p_Z \rightarrow p_G$ such that p_G and p_X are indistinguishable (or close to it)
  - What does indistinguishable entail? Measure using *f*-Divergences (i.e. KL-Divergence)

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

  - Impossible to calculate, but approximately optimizable with sampling - this is what the GAN training procedure does

AdaGAN - Greedy Approach

- Assume, given i.i.d. samples from some unknown distribution P, we can construct a simple model that outputs samples from distribution Q that approximately minimizes

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

- We can greedily train a mixture model as follows:

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

  - Choose β and Q to minimize:

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

  - It suffices to find incrementally better rather than perfectly optimal Q:

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

AdaGAN - Improvements

- Greedy fails because β decays
  - Sample from mixture will contain fewer and fewer values from respective Q

- Instead, optimize upper bound at each step to define the next model in the mixture by reweighting data
  - Under certain assumptions, the resultant mixture model can be shown to converge to the optimum

- My project - Literature review surrounding boosted GANs
  - Work prior to AdaGAN - Includes similar mixture models that are inefficient to sample to from, lead to degenerate models under certain special data distributions, etc.
  - Conditions for convergence of AdaGAN and upper bound on optimization problem
  - What assumptions and characteristics define “weak” and “strong” generators?