MATH465: FINAL PROJECT PROPOSAL
BOOSTED GENERATIVE ADVERSARIAL NETWORKS

Haiyan Wang
Duke University

Introduction

Boosting, as it was originally formulated, refers to a broad category of ensemble methods that take a meta-algorithmic approach to classification problems by aggregating many weak classifiers to create a strong classifier. This project seeks to apply the intuition and methodology underlying various boosting methods for classification to generative models. In particular, I aim to review existing approaches to boosting generative adversarial networks and potentially attempt my own modifications to those methodologies.

Generative Adversarial Networks

Generative adversarial networks (GANs) were proposed by Goodfellow et al. (2014) as a novel method for developing generative models. GANs are, as the name implies, built on an adversarial interaction between a generator and a discriminative classifier wherein the generator creates artificial samples based on some underlying distribution which are then passed to a discriminator that attempts to distinguish between true and artificial samples. More precisely, the generator G in a GAN is trained to maximize the probability that the discriminator D misclassifies artificial samples as real data. This approach strongly suggests an underlying minimax algorithm, and indeed, GANs represent a two-player minimax game in which the generator is the minimizer and the discriminator is the maximizer.

Consider a GAN in which the generator and discriminator are both multilayer perceptrons. The distribution of the generator p_g is defined by a mapping from a distribution of input noise vectors $z \sim p_z$ to generated samples $G(z, \theta_z)$ where G is a differentiable function approximated by the generator MLP with parameters θ_G. The discriminator outputs a scalar $D(x, \theta_d)$ representing the probability that x is drawn from some distribution of real data p_x rather than p_g, and it is trained by maximizing the probability that samples are correctly labeled as real or artificial. The entire GAN can then be represented by the game on the value function $V(G, D)$:

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

Both the generator and disciminator are optimized with gradient descent. Goodfellow et al. (2014) considers the theoretical case in which models have infinite capacity and training time is unlimited, though they take an iterative approach in which G and D are optimized alternately (i.e. D is optimized for k steps then G is optimized for one step, repeating until convergence). This iterative approach is adopted because optimizing D in a single loop is infeasible from a computational standpoint and would result in overfitting when training the model on a finite dataset.

Minibatch stochastic gradient descent on GANs given by Goodfellow et al. (2014)

for number of training iterations **do**

for k steps **do**

- Sample a minibatch of m noise vectors $\{z_1, ..., z_m\}$ from p_z
- Sample a minibatch of m real samples $\{x_1, ..., x_m\}$ from p_x
- Update the discriminator

$$\nabla_{\theta_D} \frac{1}{m} \sum_{i=1}^{m} [\log D(x_i) + \log(1 - D(G(z_i)))]$$

end for

- Sample a minibatch of m noise vectors $\{z_1, ..., z_m\}$ from p_z
- Update the generator

$$\nabla_{\theta_G} \frac{1}{m} \sum_{i=1}^{m} \log(1 - D(G(z_i)))$$

end for

Goodfellow et al. (2014) goes on to prove that, given the earlier assumptions of unlimited model capacity and training time, that the two-player minimax game is optimized by this algorithm and converges to the global optimum $p_g = p_x$. More broadly, there is a wealth of literature beyond the scope of this project analyzing the convergence of GANs given stricter (and more practical) bounds on the size of the models and the amount of training time as well as providing novel improvements to GAN training that improve efficiency and convergence. However, it is broadly true that the minimax game underlying GANs may not always converge to a Nash equilibrium in which both objectives are optimal (Farnia, Ozdaglar 2020).

Boosted Classification

Research into boosting algorithms arose from a question first posed by Michael Kearns in 1988 concerning the hypothesis boosting problem. The question, informally stated, asks whether 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.

The answer to Kearns' question, proven by Robert Schapire through the strength of weak learnability, is yes. Schapire demonstrated that any concept class is weakly learnable (there exists a hypothesis with slightly better than random performance) if and only if it is strongly learnable (there exists a hypothesis with arbitrarily low error). He further developed several algorithms demonstrating this finding by aggregating a number of weak classifiers with slightly better than random performance to create a strong classifier with error rate arbitrarily close to 0. The most consequential and powerful of these algorithms, Adaptive Boosting, or AdaBoost for short, was formulated along with Yoav Freund in 1995, and is outlined briefly below (drawn from my notes from Professor Cynthia Rudin's CS671 course).

AdaBoost

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

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

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

- 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

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

return H

In each iteration and for every new weak classifier, the dataset is reweighted such that samples that were previously harder to classify are more heavily weighted in the loss function of the current iteration, while samples that were easier to classify are weighted more lightly. The effect is twofold: (1) if A is deterministic, reweighting the dataset enables A to generate a new weak classifier h at each iteration even though the dataset remains the same; (2) new classifiers perform better specifically on samples that were previously misclassified frequently, improving the overall performance of the strong classifier once aggregated.

The optimization scheme underlying AdaBoost is coordinate descent on exponential loss. This loss is a function of the weak classifiers, and the weight of each weak classifier α is formulated as a direct result of this optimization method.

While most research has thus far focused on boosting classifiers, the methodologies underlying boosted classifiers like AdaBoost seem, at least intuitively, to be generalizable beyond classification tasks.

Boosted Generation

Turning now to generative tasks, I pose the question - is it possible to aggregate a number of "weak" generators to create a single "strong" generator with improved performance? Of course, in the context of generative problems, the notions of weak and strong differ substantially from their counterparts in classification, and a clearer formulation of what precisely entails weak and strong generators will be a significant focus of this project. The following section explores some of the available literature on existing approaches to boosting generators with the goal of solidifying definitions, contextualizing existing and ongoing research, and potentially discovering areas for further exploration.

AdaGAN (Tolstikhin et al., 2017)

AdaGAN is a meta-algorithmic approach to image generation that aims to address the missing modes problem prevalent in GANs wherein a generative model is unable to produce artificial examples of certain categories in the training data. AdaGAN implements an iterative procedure similar to AdaBoost in which the training dataset is repeatedly reweighted and used to train a new generator. These generators are aggregated into a mixture model, and the outputs of the combined model are used to reweight the training data for the next iteration. Though motivated originally by GANs, this methodology could be applied more broadly to any generative model.

Papers

- Kearns 88 (Hypothesis Boosting Question)
- Schapire 90 (Strength of Weak Learnability)
- Tolstikhin et al. 17 (AdaGAN)
- Goodfellow et al. 14 (GANs)
- Farnia, Ozdaglar 20 (Nash Equilibrium of GANs)