Generative Adversarial Nets

1 Citation

Goodfellow, Ian, et al. "Generative adversarial nets." Advances in neural information processing systems. 2014.

`http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf`

2 Abstract

By making a generator network G that generates examples aiming to fool a discriminator network D that aims to distinguish between real and fake examples, we can train powerful generative models.

3 Introduction

It has been hard to train deep generative models. Generative Adversarial Nets (GANs) can be trained with regular backpropagation and dropout.

4 Related Work

Prior work on deep models makes them estimate the parameters of a probability distribution, which enables us to maximize the log-likelihood of the data. Restricted Boltzmann Machines (RBMs) and Generative Stochastic Networks are examples of such models. Variational Autoencoders (VAE), like GANs, have a generative model, but they use a recognition model instead of a discriminator model. VAEs can model binary outputs but cannot model binary latent variables - GANs are the opposite. Noise Contrastive Estimation (NCE) uses a discriminative model to distinguish between noise and the real dataset. If the noise distribution is actually a generative model, you can use this framework to train successively better generative models. Predictability minimization also makes two networks compete, but our approach is a more general competitive framework. A GAN is a minimax game where the generator and discriminator push an objective function in opposite directions. GANs are not the same thing as adversarial examples, which are inputs designed to make a neural net predict the wrong class output.

5 Adversarial Nets

The input to the generator is noise vector $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})$. Our minimax game is:

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

We optimize this function by training D for k steps, training G for one step, and repeating. Early in training, G is poor so $\log(1 - D(G(\boldsymbol{z})))$ saturates. Thus, at the start of training, G instead maximizes $\log D(G(\boldsymbol{z}))$

6 Theoretical Results

For a fixed G, the optimal discriminator is:

$$D_G^*(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})} \tag{2}$$

The value $C(G) = \max_D V(G, D)$ is maximized when $p_g = p_{data}$. If our model is sufficiently high capacity, our algorithm will converge.

7 Experiments

We train GANs on MNIST, Toronto Faces, and CIFAR-10. The generator used ReLU and sigmoid activations while the discriminator uses maxout activations. We use dropout for training the discriminator. The generator's input is noise. Our generator can create examples, but can't compute the probability density of those examples. We fit a Gaussian Parzen Window to the generator outputs for this purpose. It doesn't work great in high dimensional spaces and has high variance, but it's the best we can do.

8 Advantages and Disadvantages

There's no explicit representation of $p_g(\boldsymbol{x})$. It can be tricky to train the generator and discriminator in sync. To avoid a situation where G collapses a bunch of different $\boldsymbol{z}$ values into a single $\boldsymbol{x}$, we need to train D for k steps each time we train G for one step. However, GANs are much more computationally cheaper than Markov Chain Monte Carlo and they can represent spiky distributions.

9 Conclusions and Future Work

You can a conditional generative model (i.e. one that generates inputs based on vector $\boldsymbol{c}$). You could train a network to predict $\boldsymbol{z}$ given $\boldsymbol{x}$. Maybe the features learned from the discriminator and generator could help in supervised learning tasks where the dataset is small.