Generative Adversarial Networks

May 20, 2019

Discriminative models vs. Generative models

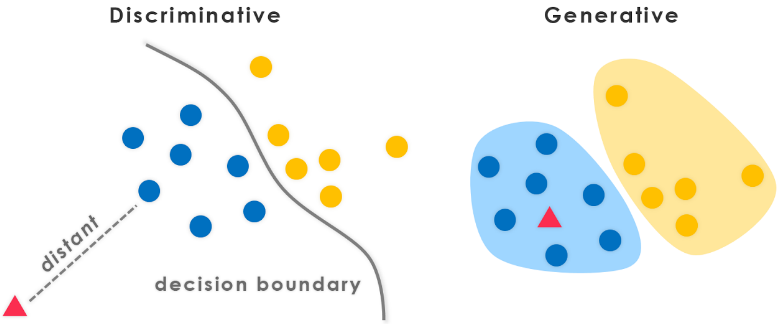

Figure: Reprinted from www.evolvingai.org/fooling.

Discriminative models

- Here, we want to find (or approximate) $p(y|x)$.
- y, often called the *label*, can be thought as another feature.
- Given a set of features x, we want to predict a certain feature, y.
- Given the pixel values of an image, how likely is it to contain a dog?
- You can see that we can only hope to learn the relation between x and y.
- If we train a classifier to classify the images of cats and dogs, it only learns necessary boundaries to achieve its goal (minimize the loss).

Discriminative models

Discriminative models are very successful when you have lots of data with lots of classes and a large model.

- lots of data $\implies$ $\mathbb{R}^d$ is covered with a lot examples therefore reducing the possibility of finding trivial solutions.
- lots of classes $\implies$ we need lots of boundaries, *hyperplanes*, to discriminate different classes. Again, reducing the possibility of finding trivial solutions.
- a large model $\implies$ we have lots of hyperplanes.

Generative models

- Here, we want to find (or approximate) $p(x, y)$.
- Given a set of features (x and y), we want to know how likely it is to be from our data distribution.
- Since we care about the likelihood of a sample, we have to learn the relation between all features, not only x and y.
- These models are called generative models since we have a likelihood function, we can simulate the data generating distribution.
- It is not very easy though.
- $p(y|x)$ comes for free since we know the joint distribution, $p(x, y)$.

Generative Adversarial Networks

- People are interested in generative models (it is superior).
- Although deep learning has achieved great success on discriminative models, that is not the case for generative models
- The work "Generative adversarial nets (Goodfellow et al. 2014)" is an attempt to remedy this gap.
- Also, "Autoencoding Variational Bayes (Kingma and Welling 2013)" a.k.a. Variational Autoencoders, VAE.

Generative Adversarial Networks

- We want to estimate $p(x)$.
- If we have $x \in p(x)$ and $\bar{x} \notin p(x)$, we can train a classifier for two-class classification.
- We have $x \in p(x)$ (our data itself), but not $\bar{x} \notin p(x)$ (complement of our data?).
- You might say why don't you just generate $\bar{x}$ randomly. It is almost impossible to generate the whole state space randomly.
- Also, we do not need every possible configuration, just the ones that are near $p(x)$.

Generative Adversarial Networks

- Let's have a neural net D_ϕ, that is parameterized by ϕ and a neural net G_θ that is parameterized by θ.
- Given an arbitrary noise distribution, G tries to produce samples that looks like samples from $p(x)$.
- For $z \sim p(z)$, $\bar{x} = G(z)$. $p(z)$ can be anything but mostly chosen as Gaussian.
- D tries to make a two-class classification, outputs 1 when the data is real 0 when the data is generated.

$$\max \quad r \log D(x) + (1-r)\log(1-D(x)) \tag{1}$$

$$\max \quad \mathbb{E}_{x \sim p(x)}[\log D(x)] + \mathbb{E}_{z \sim p(z)}[log(1 - D(G(z))] \tag{2}$$

Generative Adversarial Networks

- While we optimize D, we also optimize G to produce better samples by minimizing the objective (the objective which D tries to maximize).

$$\min_G \max_D \quad \mathbb{E}_{x \sim p(x)}[\log D(x)] + \mathbb{E}_{z \sim p(z)}[log(1 - D(G(z))] \tag{3}$$

- This corresponds to a minimax game where the optimal point is a saddle point.
- Notice that G is trained with the error that is backpropagated through D.

Generative Adversarial Networks

"G can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while D is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles"
- Goodfellow et al. 2014

Current problems with GANs

You end up with a D that can be thought as a generative model in a sense (since it learns data distribution) and a sampling function G that is easy to sample from. It is a nice idea but there are problems.

- Mode drop: G only produces some partition of $p(x)$.
- Mode collapse: G only produces the same sample.
- Vanishing or exploding gradients: Since we train G with the feedback of D, instead of a nice convex loss function, this can result in undesired behaviors.

But if you train it well...

A neural network can generate samples.

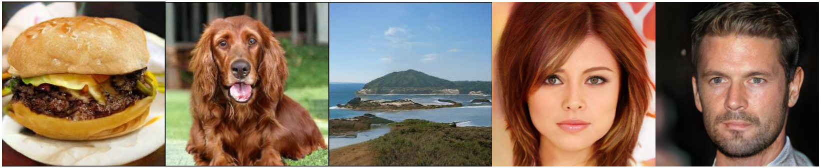

Figure: Reprinted from Brock et al. 2018 and Karras et al. 2017

Figure: Reprinted from Karras et al. 2017

Even tries to generate memes...

Figure: Reprinted from Karras et al. 2017

References

"Generative Adversarial Nets" Goodfellow et al. 2014
"Autoencoding Variational Bayes" Kingma and Welling 2013
"Progressive Growing of GANs for Improved Quality, Stability, and Variation" Karras et al. 2017
"Large Scale GAN Training for High Fidelity Natural Image Synthesis" Brock et al. 2018

For those who are interested in GANs: ahmetoglu.alper@gmail.com