Generative Adversarial Networks (GANs)

The coolest idea in Machine Learning in the last twenty years - Yann Lecun

Introduction

Supervised Learning  Unsupervised Learning
Semi-supervised Learning
Reinforcement Learning

From David silver, Reinforcement learning (UCL course on RL, 2015).
Supervised Learning

- Find deterministic function \( f: \ y = f(x), \ x: \text{data}, \ y: \text{label} \)

Unsupervised Learning

- More challenging than supervised learning. No label or curriculum.
- Some NN solutions:
  - Boltzmann machine
  - AutoEncoder
  - Generative Adversarial Networks

\[ [0.1, 0.3, -0.8, 0.4, \ldots] \xrightarrow{g} \]

"Most of human and animal learning is unsupervised learning. If intelligence was a cake, unsupervised learning would be the cake, supervised learning would be the icing on the cake, and reinforcement learning would be the cherry on the cake. We know how to make the icing and the cherry, but we do not know how to make the cake. We need to solve the unsupervised learning problem before we can even think of getting to true AI." - Yann Lecun

"What I cannot create, I do not understand" - Richard P. Feynman

Generative Models

Introduction

- What does "generative" mean?
  - You can sample from the model and the distribution of samples approximates the distribution of true data points
- To build a sampler, do we need to optimize the log-likelihood?
- Does this sampler work?

```python
import glob, cv2
files = glob.glob('*.jpg')
def _sample():
    idx = np.random.randint(len(files))
    return cv2.imread(files[idx])
def sample(*, n_samples):
    samples = np.array([_sample() for _ in range(n_samples)])
    return samples
```
Generative Adversarial Networks

Desired Properties of the Sampler

- What is wrong with the sampler? Why is not desired?

Desired Properties
- We don’t want to sample the same existing data points
- Build a generative model that understands structure and underlying distribution of data points
- Output similar but not the same points in the training dataset
- Example: Faces with different poses, Different handwriting etc

Generative Adversarial Networks (GANs)

Implicit Models

- Given samples from data distribution $p_{data} : x_1, x_2, \ldots, x_n$
- Given a sampler $q_\phi(z) = DNN(z; \phi)$ where $z \sim p(z)$
- Let $p_{model}$ be the density function of $x = q_\phi(z)$
- We do not have an explicit form for $p_{data}$ or $p_{model}$
- Objective: Make $p_{model}$ as close as possible to $p_{data}$ by learning $\phi$
Generative Adversarial Networks
Moment Matching

- Given samples $X = x_1, x_2, \ldots, x_n$ from $P_X$ and samples $Y = y_1, y_2, \ldots, y_n$ from $P_Y$, how do we compare $P_X$ and $P_Y$?
- Match moments of both $P_X$ and $P_Y$ to bring them closer
- Not feasible to compute higher order moments in higher dimensions

$$L_{MMD} = \left\| \frac{1}{N} \sum_{i=1}^{N} \phi(x_i) - \frac{1}{M} \sum_{j=1}^{M} \phi(y_j) \right\|^2$$

\[= \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} \phi(x_i)^T \phi(x_{i'}) + \frac{1}{M^2} \sum_{j=1}^{M} \sum_{j'=1}^{M} \phi(y_j)^T \phi(y_{j'}) - \frac{2}{MN} \sum_{i=1}^{N} \sum_{j=1}^{M} \phi(x_i)^T \phi(y_j)\]

\[= \frac{1}{N^2} \sum_{i=1}^{N} \sum_{i'=1}^{N} k(x_i, x_{i'}) - \frac{2}{MN} \sum_{i=1}^{N} \sum_{j=1}^{M} k(x_i, y_j) + \frac{1}{M^2} \sum_{j=1}^{M} \sum_{j'=1}^{M} k(y_j, y_{j'})\]
Generative Adversarial Networks (GANs)

Value of Expectation prob. of D(\text{real}) prob. of D(\text{fake})

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

Minimize G Maximize D

x is sampled from real data z is sampled from N(0, I)
Generative Adversarial Networks (GANs)

What is going on?

- What is the optimal discriminator given generated and true distributions?

\[
V(G, D) = \mathbb{E}_{x \sim p_{\text{data}}} \left[ \log D(x) \right] + \mathbb{E}_{z \sim p(z)} \left[ \log(1 - D(G(z))) \right]
\]

\[
= \int_x p_{\text{data}}(x) \log D(x) dx + \int_z p(z) \log(1 - D(G(z))) dz
\]

\[
= \int_x p_{\text{data}}(x) \log D(x) dx + \int_x p_{g}(x) \log(1 - D(x)) dx
\]

\[
= \int_x \left[ p_{\text{data}}(x) \log D(x) + p_{g}(x) \log(1 - D(x)) \right] dx
\]

\[
\nabla_y [a \log y + b \log(1 - y)] = 0 \implies y^* = \frac{a}{a + b} \quad \forall \quad [a, b] \in \mathbb{R}^2 \setminus [0, 0]
\]

\[
\implies D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_{g}(x)}
\]
What is the objective of the generator given the optimal discriminator?

\[
V(G, D^*) = \mathbb{E}_{x \sim p_{data}} [\log D^*(x)] + \mathbb{E}_{x \sim p_g} [\log (1 - D^*(x))]
\]

\[
= \mathbb{E}_{x \sim p_{data}} \left[ \log \frac{p_{data}(x)}{p_{data}(x) + p_g(x)} \right] + \mathbb{E}_{x \sim p_g} \left[ \log \frac{p_g(x)}{p_{data}(x) + p_g(x)} \right]
\]

\[
= -\log(4) + KL \left( p_{data} \parallel \frac{p_{data} + p_g}{2} \right) + KL \left( p_g \parallel \frac{p_{data} + p_g}{2} \right)
\]

(Jensen-Shannon Divergence (JSD) of \( p_{data} \) and \( p_g \) \( \geq 0 \))

\[
V(G^*, D^*) = -\log(4) \text{ when } p_g = p_{data}
\]
Generative Adversarial Networks

Discriminator Saturation

\[ \nabla_{G(z)} \log(1 - D(G(z))) \text{ where } D(x) = \text{sigmoid}(x; \theta) = \sigma(x; \theta) \]

\[ \nabla_x \sigma(x) = \sigma(x)(1 - \sigma(x)) \]

---

Generative Adversarial Networks

Optimization

**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.

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

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

(Taken from Goodfellow et al 2014)
Generative Adversarial Networks

What is going on?

Generative Adversarial Networks (GANs)

Figure 1: Generative adversarial nets are trained by simultaneously updating the discriminative distribution ($D$, blue, dashed line) so that it discriminates between samples from the data generating distribution (black, dotted line) $p_{\text{data}}$ from those of the generative distribution $p_G$ ($G$, green, solid line). The lower horizontal line is the domain from which $z$ is sampled, in this case uniformly. The horizontal line above is part of the domain of $x$. The upward arrows show how the mapping $x = G(z)$ imposes the non-uniform distribution $p_G$ on transformed samples. $G$ contracts in regions of high density and expands in regions of low density of $p_G$. (a) Consider an adversarial pair near convergence: $p_G$ is similar to $p_{\text{data}}$ and $D$ is a partially accurate classifier. (b) In the inner loop of the algorithm $D$ is trained to discriminate samples from data, converging to $D^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}$. (c) After an update to $G$, gradient of $D$ has guided $G(z)$ to flow to regions that are more likely to be classified as data. (d) After several steps of training, if $G$ and $D$ have enough capacity, they will reach a point at which both cannot improve because $p_G = p_{\text{data}}$. The discriminator is unable to differentiate between the two distributions, i.e. $D(x) = \frac{1}{2}$.

Generative Adversarial Networks (GANs)

Live Action

- GAN LAB
Generative Adversarial Networks

DCGAN Results
Evaluation of GANs

- Still an open research problem
- Some Measures:
  - Inception Score
  - Frechet Inception Distance

Inception Score

- How do we say generated images are good?
- Good generators generate semantically diverse samples
- Use a trained model (Inception v3) to model $p(y | x)$
  - Each image should have a recognizable object from a class $\rightarrow p(y | x)$ should have low entropy
  - There should be many classes generated $\rightarrow p(y)$ should have high entropy
Inception Score

\[ IS(x) = \exp(\mathbb{E}_{x \sim p_g} [D_{KL} [p(y|x) \parallel p(y)]] \]
\[ = \exp(\mathbb{E}_{x \sim p_g, y \sim p(y|x)} [\log p(y|x) - \log p(y)]) \]
\[ = \exp(H(y) - H(y|x)) \]

What GAN you do?
What GAN you do?

<table>
<thead>
<tr>
<th>Text description</th>
<th>Stage-I images</th>
<th>Stage-II images</th>
</tr>
</thead>
<tbody>
<tr>
<td>This bird is blue with white and has a very short beak</td>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>This bird has wings that are brown and has a yellow belly</td>
<td><img src="image3.png" alt="Image" /></td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>A white bird with a black crown and yellow beak</td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td>This bird is white, black, and brown in color, with a brown beak</td>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
</tr>
<tr>
<td>The bird has small beak, with reddish brown crown and gray belly</td>
<td><img src="image9.png" alt="Image" /></td>
<td><img src="image10.png" alt="Image" /></td>
</tr>
<tr>
<td>This is a small, black bird with a white breast and white on the wingbars.</td>
<td><img src="image11.png" alt="Image" /></td>
<td><img src="image12.png" alt="Image" /></td>
</tr>
<tr>
<td>This bird is white black and yellow in color, with a short black beak</td>
<td><img src="image13.png" alt="Image" /></td>
<td><img src="image14.png" alt="Image" /></td>
</tr>
</tbody>
</table>

References

- CS294-158 Deep Unsupervised Learning Spring 2019 - UC Berkeley
- CS231n Deep Learning for Computer Vision - Stanford University
- Tutorial on GANs, Ian Goodfellow