Lecture 18

Generative Adversarial Networks

Idea: train 2 networks - G generator and D discriminator

\[ \text{noise } z \rightarrow \text{data space } x \]

\[ z \sim p(z) \]

\[ z \in \mathbb{R}^n \]

\[ n \text{ is size of data space} \]

\[ x \in \mathbb{R}^d \]

WANT: \( x \sim p_{\text{data}}(x) \)

D (classifier)

\( D(x \text{ or } G(z)) \rightarrow \text{class probability} \)

tries to distinguish real from fake

want G to fool D

and D to be as good as possible
loss function: \[ L = \sum_{i \in \text{data}} \log \Theta(x_i) + \sum_{i \in \text{gen}} \log (1 - \Theta(G(x_i))) \]

usual BCE but negative

\[ \min_{G} \max_{\Theta} L \rightarrow \text{saddle point optimization} \]

unstable could be multiple saddles
convergence is tricky

discriminator is a good classifier

\[ \hat{G}(x) = \frac{\text{P}_{\text{data}}(x)}{\text{P}_{\text{data}}(x) + \text{P}_{\hat{G}}(x)} \]

plug back into \( L \)

\[ \sum_{x \in \text{data}} \log \frac{\text{P}_{\text{data}}(x)}{\text{P}_{\text{data}}(x) + \text{P}_{\hat{G}}(x)} + \sum_{x \in \text{gen}} \log \frac{\text{P}_{\hat{G}}(x)}{\text{P}_{\text{data}}(x) + \text{P}_{\hat{G}}(x)} \]
\[
\max_D \mathbb{E}_x \left[ \log \frac{p_{\text{data}}(x)}{p_G(x)} \right] + \mathbb{E}_z \left[ \log \frac{p_G(z)}{q(z)} \right] = \mathbb{E}_x \left[ \log \frac{p_{\text{data}}(x)}{p_G(x)} \right] = \text{const}
\]

\[
\min_G \left( \mathbb{E}_x \left[ \log \frac{1}{p_G(x)} \right] + \mathbb{E}_z \left[ \log \frac{1}{q(z)} \right] \right) \rightarrow \left\{ \begin{array}{l}
0 \leq \text{JS}(D) \leq 1 \\
\text{JS}(D) = 0 \iff p_{\text{data}} = p_G
\end{array} \right.
\]

GAN learns \( p_{\text{data}}(x) \) implicitly!

Example:

"likelihood ratio tricks" + "likelihood-free inference learning"

We don't ever get \( p_{\text{data}}(x) \) itself. Given \( x \) we can't know value of \( p_{\text{data}}(x) \). We only get samples.
Training a GAN:

1. Take a batch from real data \([x]\)
2. Generate data \([x']\)
3. Train discriminator
4. Freeze discriminator
5. Take sample from generated data
6. Train generator
7. Repeat