Universal Approximation Theorem

Tips

- GANs are sensitive to hyperparameters
- The generator wants to **minimise log(1 - D(g(z)))**, but it's easier to do **maximising log(D(G(z))**, because it results in better training, as minimising a value can lead to less or no training

GANs - a combination of **two** models, the **generative** model and the **discriminative** model

- We create new data points using the generator, in order to train the discriminative model
    - We then make the discriminative model learn whether it is original or generated
    - **Adversarial setup -** they compete against each other
        - **Minimax Game -** the **generator** is trying to **minimise** the performance of the **discriminator**, whereas the **discriminator** is trying to **maximise** his score

$$E(\mathcal{L}) = E\left(\ln[D(x)]\right) + E\left(\ln[1 - D(G(z))]\right)$$

$$\sum p_{data}(x) \ln[D(x)] + \sum p_z(z) \ln[1 - D(G(z))]$$

$$\int p_{data}(x) \ln[D(x)]\, dx + \int p_z(z) \ln[1 - D(G(z))]\, dz$$

- **Loss Function -** we use the **binary cross entropy** loss function
    - This is the same as the **binary cross entropy** loss function, however, the y and y^ are replaced with y = 1 or 0, and y^ = D(X) or D(G(Z))
    - The **E** is the expectation (the average) score
    - When data is **discrete**, we **sum over the score * the probability**
    - When data is **continuous**, we define an **integral over the score * the probability of the data**
    - However, we just use **E** to simplify the above two cases

- **Optimising the Loss Function**
    - We first **fix the learning of G**
    - We then use **gradient descent to optimise the discriminator D weights** using a dataset formed of the original data and the generated data
    - We then **fix the learning of D**
    - We then generate data samples and use **gradient descent to optimise generator G weights**
        - *The partial derivative of the original data case w.r.t G is equal to 0*
    - For every **k** updates of the **discriminator**, we are only updating the **generator once**

update θ_d by grad. ascent

$$\frac{\partial}{\partial \theta_d} \frac{1}{m}\left[\ln\left[D(x)\right] + \ln\left[1-D(G(z))\right]\right]$$

$$\frac{\partial}{\partial \theta_g} \frac{1}{m}\left[\ln\left[1-D(G(z))\right]\right]$$

Generative Model - Learns the joint probability P(X, Y)

- **P(X, Y) = P(X|Y) P(Y) = P(Y|X) P(X)**
- Most common implementation is **Naive Bayes**
- We use the generative model to **create new instances of data**
- This is because we learn the distribution function of the data itself
- The generative model takes **noise as input**
    - You sample the noise using some distribution, i.e. normal distribution
- **Has 1 input vector** (noise)

Discriminative Model - Learns the conditional probability P(Y|X = x)

- Used when we want to predict the class of some input data
- The **discriminator** computes the probability that the input is original / generated
- Generally is a simple **binary classifier**
- We can put the output of the discriminator through an **activation function** (eg sigmoid) to give us a **% prob.**
- **Has 1 output vector**

Proof that the probability distributions of P(G(Z)) = P(X) at the Global Minimum

- We **fix G**
- We then try to find for which value of the **discriminator D**, the value of our **loss function** is the **minimum**
- We do this by finding its **derivative**
- We then **fix D**, then substitute the derivative of D into the loss function
- We now want to find the point where our **loss function** is the **maximum** (biggest loss)
- *We then use **JS divergence** to calculate the difference between two probability distributions*
    - We take the formula for JS divergence, and try to go from V -> JS(P(X), P(G(Z))
    - *By multiplying the parts inside the brackets of ln in our expression for V, we can obtain JS*
    - We discover a term **-2ln2** (which is the minimum value of V (min of our score, max loss))
    - JS(...) = 0 only when $P_{data} = P_g$
    - This means at the global minimum of our value function, $\mathbf{P_{data} = P_g}$

For fixed G,

$$V(G,D) = \int_x p_{data}(x)\ln[D(x)] + p_g(x)\ln[1-D(x)]\ dx$$

will be maximum for $D(x) =$

$$\frac{p_{data}(x)}{p_{data}(x) + p_g(x)}$$

$$\min_G V = E_{x\sim p_{data}} \ln\left(\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right) + E_{x\sim p_g} \ln\left(1 - \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right)$$

$$= E_{x\sim p_{data}} \ln\left(\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right) + E_{x\sim p_g} \ln\left(\frac{p_g(x)}{p_{data}(x)+p_g(x)}\right)$$

$$JS(p_1 \| p_2) = \frac{1}{2} E_{x\sim p_1} \ln\left(\frac{p_1}{\frac{p_1+p_2}{2}}\right) + \frac{1}{2} E_{x\sim p_2} \left(\frac{p_2}{\frac{p_1+p_2}{2}}\right)$$

$$\min_G V = E_{x\sim p_{data}} \ln\left(\frac{p_{data}(x)}{\frac{p_{data}(x)+p_g(x)}{2}}\right) + E_{x\sim p_g} \ln\left(\frac{p_g(x)}{\frac{p_{data}(x)+p_g(x)}{2}}\right) - 2\ln 2$$

$$\min_G V = 2\, JS(p_{data} \| p_g) - 2\ln 2$$