CONDITIONAL GAN SUMMARY WEEK3

SMITALI BHANDARI

First let's discuss about GANs- *Generative Adversial Networks* in brief.

- This framework consists of two models, Generative and Discriminator. The Generative model G counterfeits a probability distribution, whereas the discriminator D learns to determine whether a sample is from a true data set or is generated.
- First we define the generator's distribution p_g over data x, by initializing it with input noise variables $p_z(Z)$(usually a Gaussian or standard probability distribution). Then we define $G(z, \theta_g)$, where G is differential function(a multilayer neural network) with parameters θ_g. $G(z, \theta_g)$ transforms noise z into a synthetic probability distribution. It learns through back propagation.
- Next, $D(\mathbf{x}; \theta_d)$ is another multilayer neural network, where $D(x)$ represents the probability that $\mathbf{x}$ came from the true data rather than the generated p_g. It is clear that $D(\mathbf{x}; \theta_d)$ outputs a single scalar, whereas G's output was a probability distribution.
- We train D to maximize the probability of assigning correct label, G to minimize. In other words, we train G to minimize the loss function and D to maximize. The loss function is given by:

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

- Thus, this is similar to a two player min max game with the above value function.
- There were three major results discussed in the GANs paper:

Proposition 1. *For fixed G, the optimal discriminator D is*

$$D^*_G(\boldsymbol{x}) = \frac{p_{data}(\boldsymbol{x})}{p_{data}(\boldsymbol{x}) + p_g(\boldsymbol{x})}$$

For the proof: Optimal discriminator implies to maximize the quantity $V(G, D)$. So we basically bring the equation in terms of $\boldsymbol{x}$ and then differentiate w.r.t. $\boldsymbol{x}$ to see where it is at its maxima.

Theorem 2. *The global minimum of training $C(G) = maxV(G, D)$ is achieved iff $p_g = p_data$. At this point $C(G) = -log4$*
For the proof: Write $C(G)$ in KL divergence terms. The expression turns out to be same as Shannon divergence between model's distribution and data generator.

$$C(G) = -log(4) + 2JSD(p_data||p_g)$$

Shannon divergence b/n two distributions is always non-negative and is zero(min) when they are equal. Hence proved!

Thus, the basic idea is to train the models using back propagation so as to find nash equilibrium for the two player min-max game.

Conditional Generative Adversial Network:

- GANs are extended such that, both the generator and discriminator are conditioned on extra information $\mathbf{y}$, which could be any kind of information such as class labels or any other modalities. $\mathbf{y}$ will be fed in as extra input layer.
- For generator, $p_z(z)$ and $\mathbf{y}$ are combined in joint hidden representation. Whereas, in discriminator, $\mathbf{x}$ and $\mathbf{y}$ are seperately fed as input to a disriminative function(MLP).
- Two player minimax game:

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

-
  - **Unimodal**
    CGAN was trained on MNIST images conditioned on their class labels with one hot encoding. The following table shows the details of generator model, where the outpul layer is final sigmoidal layer.

Step	**Input Dimension**	**Number of Nodes in Layer**
Noise Input ($\mathbf{z}$)	100D vector	-
First Hidden Layer	100D $\to$ 200D	200 nodes
Second Hidden Layer	200D $\to$ 1000D	1000 nodes
Third Hidden Layer	1000D $\to$ 1200D	1200 nodes
Output Layer	1200D $\to$ 784D	784 nodes (MNIST pixels)

Table 1. Generator net

For the discriminator:

Step	**Input Dimension**	**Number of Units / Pieces**
Input ($\mathbf{x}$)	-	-
First Hidden Layer (Maxout)	$\mathbf{x} \to 240D$	240 units, 5 pieces
Second Hidden Layer (Maxout)	$\mathbf{y} \to 50D$	50 units, 5 pieces
Joint Hidden Layer (Maxout)	$(240D, 50D) \to 240D$	240 units, 4 pieces
Output Layer (Sigmoid)	$240D \to 1$	1 (Binary classification)

Table 2. Discriminator

Maxout function is a type of neural network activation function. A neuron (unit) in a layer learns multiple linear transformations(pieces) and outputs the maximum as output of that unit.
Model training: Using stochastic gradient decent with mini-batches of size 100, initial learning rate of 0.1, with decay factor of 1.00004. Moment with initial value of .5. Dropout[9] with probability 0.5 was applied to both G and D models.
The results outperformed several approaches including normal GANs.

- **Multimodal** the paper demonstrated automated tagging of images, with multi-label predictions using CGANs to generate distibution of tag-vectors conditional on image features.
  Model details:

Feature Type	Method Used
Image Features	Pre-trained convolutional model on ImageNet dataset (21,000 labels)
Text Features	Skip-gram model trained on YFCC100M dataset metadata (titles, tags, descriptions)
Word Vector Size	200
Vocabulary Size	247,465 (words appearing ¡200 times omitted)

Table 3. Image and Text Feature Extraction

Aspect	Details
Dataset Used	MIR Flickr 25,000
Training Set Size	150,000 examples
Image Tags Processing	Images with multiple tags repeated in training
Evaluation Method	100 samples per image, top 20 closest words selected based on cosine similarity
Final Tag Selection	Top 10 most common words among the 100 samples

Table 4. Dataset and Training Setup

Component	Architecture
Generator	Gaussian noise (size 100) → 500-dimension ReLU layer → 4096-dimension image feature vector → 2000-dimension ReLU hidden layer → 200-dimension linear layer (word vectors)
Discriminator	500 & 1200-dimension ReLU layers → Maxout layer (1000 units) → Join layer (3 pieces) → Sigmoid output
Training Algorithm	Stochastic Gradient Descent (SGD) with mini-batches of 100
Learning Rate	Initial: 0.1, exponentially decreased to 0.000001
Momentum	Initial: 0.5, increased to 0.7
Dropout	Applied to generator & discriminator (probability: 0.5)
Hyperparameter Selection	Cross-validation and random grid search

Table 5. Model Architecture and Training Details

Acknowledgement

I would like to express my gratitude to GPT version 4.0 for the tables for multimodal model details.