Assignment 1
Machine Learning 2

Debprasad Kundu

Roll No. CS2102

MTech in Computer Science

Contents

1 Problems on Autoencoder

1.1 Solution 1

- **Denoising autoencoders** : Denoising autoencoders are an extension of simple autoencoders; however, it's worth noting that denoising autoencoders were not originally meant to automatically denoise an image.

  Autoencoders are Neural Networks which are commonly used for feature selection and extraction. However, when there are more nodes in the hidden layer than there are inputs, the Network is risking to learn the so-called "Identity Function", also called "Null Function", meaning that the output equals the input, marking the Autoencoder useless. Denoising Autoencoders solve this problem by corrupting the data on purpose by randomly turning some of the input values to zero.

- **Pseudocode for Denoising Autoencoder training loop :**

  $\mathbf{x} = [x_1, x_2, ...x_n] \in R^{n*m}$ is the input matrix, in which $x_i \in [0,1]^m$ $(1 \leq i \leq m)$ is a single input data

  e is the amount of epochs to be iterated

  b is the amount of batches

  l is the learning rate

  c is the corruption level

  $\theta = \{W, \mathbf{b}, \mathbf{b_n}\}$ where $W \in R^{n*d}$, $\mathbf{b} \in R^d$, $\mathbf{b_n} \in R^d$, θ is the parameters of a Denoising Autoencoders.

  Algorithm :

for *0 to e* **do**
  for *0 to b* **do**
    $\tilde{\mathbf{x}}$ = getCorruptedInput($\mathbf{x}$), in which c is the corrupted level
    $\mathbf{h}$ = sigmoid($\tilde{\mathbf{x}} * W + \mathbf{b}$)
    $\hat{\mathbf{x}}$ = sigmoid($\mathbf{h} * W^T + \mathbf{b_n}$)
    $\mathbf{L}(x, \hat{\mathbf{x}})$ = -$\sum_{i=0}^{d}[\mathbf{x}_i log\hat{\mathbf{x}}_i + (1 - \mathbf{x}_i)log(1 - \mathbf{x}_i)]$
    cost = mean($\mathbf{L}(x, \hat{\mathbf{x}})$)
    $\mathbf{g}$ = compute the gradients of the cost with respect to θ
    for θ_i, g_i *in* (θ, $\boldsymbol{g}$) **do**
      $\theta_i = \theta_i - l * g_i$
    end
  end
end

1.2 Solution 2 (Coding)

Code for Denoising Autoencoder

2 Problems on Generative Adversarial Networks

2.1 Solution 1

1. Yes, the use of BCE loss lead to the unstable training of GAN by being saturated.

   Yes, LSGANs is the improved stability of learning process.

2. **First Part:**

Objective function of LSGAN:

Let the GAN consisting of Generator **G** and a discriminator **D**. Let us take the original data distribution as p_{data}, that generated by **G** as p_g and that of the noise as p_z. Then the LS loss objective function is given by,

$$min_D = \frac{1}{2}E_{x\sim p_{data}}(\mathbf{D}(x)-b)^2 + \frac{1}{2}E_{z\sim p_z}(\mathbf{D}(\mathbf{G}(z))-a)^2 \tag{1}$$

$$min_G = \frac{1}{2}E_{x\sim p_z}(\mathbf{D}(\mathbf{G}(z))-c)^2 \tag{2}$$

where a, b and c respectively denotes the generated data label, the real data label and the label of the data which the generator wants the discriminator to believe.

Second Part:

Let,

$$f(\mathbf{D}) = \frac{1}{2}E_{x\sim p_{data}}(\mathbf{D}(x)-b)^2 + \frac{1}{2}E_{z\sim p_z}(\mathbf{D}(\mathbf{G}(z))-a)^2 \tag{3}$$

Equation 3 can be written as

$$f(\mathbf{D}) = \frac{1}{2}\int_x p_{data}(x)(\mathbf{D}(x)-b)^2 + p_z(x)(\mathbf{D}(x)-a)^2 dx \tag{4}$$

Now minimizing $f(\mathbf{D})$ is equivalent to minimizing $p_{data}(\mathbf{D}(x)-b)^2 + P_G(\mathbf{D}(x)-a)^2$. So let,

$$L(\mathbf{D}) = p_{data}(x)(\mathbf{D}(x)-b)^2 + p_g(x)(\mathbf{D}(x)-a)^2 \tag{5}$$

Now, $\frac{\delta L}{\delta D} = 2p_{data}(x)(\mathbf{D}(x)-b) + 2p_g(x)(\mathbf{D}(x)-a) = 0$

$\Rightarrow \boxed{\mathbf{D}^*(x) = \frac{bp_{data}(x)+ap_g(x)}{p_{data}(x)+p_g(x)}}$

Then put the value of $\mathbf{D}^*$ in $\frac{1}{2}E_{x\sim p_{data}}(\mathbf{D}(x)-c)^2 + \frac{1}{2}E_{x\sim p_g}(\mathbf{D}(x)-c)^2$ we have,

$2C = \int_x [p_{data}(x)(\frac{bp_{data}(x)+ap_g(x)}{p_{data}(x)+p_g} - c)^2 + p_g(x)(\frac{bp_{data}(x)+ap_g(x)}{p_{data}(x)} - c)^2]dx$

$= \int_x \frac{((b-c)(p_{data}(x)+p_g(x))-(b-a)p_g(x))^2}{p_{data}(x)+p_g(x)}dx$

If we take $b-c=1$ and $b-a=2$,then

$\boxed{2C = \chi^2_{Pearson}(p_{data}+p_g||2p_g)}$

2.2 Solution 2 (Coding)

Code for GAN

2.3 Solution 3 (Coding)

Code for DCGAN

3 Problems on Variational Auto-Encoder

3.1 Solution 1 (Coding)

Variational Auto-Encoder

4 Questioning the quality of the generated samples

4.1 Solution 1

Inception Score : The Inception Score is based on a heuristic that realistic samples should be able to be classified when passed through a pre-trained network, such as Inception on ImageNet. Besides high predictability (low entropy), the Inception Score also evaluates a GAN based on how diverse the generated samples are (e.g. high variance or entropy over the distribution of generated samples). This means that there should not be any dominating classes.

$$IS=\exp\left(E_{x\sim p_g} D_{KL}(p(y|x)||p(y))\right) \tag{6}$$

where p_g be the distribution of generated data.

Fréchet Inception Distance : FID estimates realism by measuring the distance between the generated distribution of images and the true distribution. FID embeds a set of generated samples into a feature space given by a specific layer of Inception Net. This embedding layer is viewed as as a continuous multivariate Gaussian, then the mean and covariance are estimated for both the generated data and the real data. The Fréchet distance between these two Gaussians is then used to quantify the quality of generated samples. A lower FID corresponds to more similar real and generated samples.

$$IS=\exp\left(E_{x\sim p_g} D_{KL}(p(y|x)||p(y))\right) \tag{7}$$