SAIDL Summer Challenge 2024- Computer Vision

- Aishwarya Naidu

Task

To implement a variational autoencoder(VAE) trained on the MNIST dataset and compare generations sampled from different kinds of distributions. Conventionally VAEs are assumed to have a normally distributed prior. The challenge lies in extending VAEs to the case where the latent space is assumed to have other kinds of distributions - such as beta or gamma.

Terminologies used in this report

Regular VAE: where the prior distribution is assumed to be standard normal: N~(0,1), as described by [1312.6114] Auto-Encoding Variational Bayes (arxiv.org)

Gaussian(1,2) VAE: Where the prior distribution is assumed to be a Gaussian: N~(1,2), as given in the problem statement

Beta (1,1) VAE: Where the prior distribution is assumed to be a Beta distribution, as given in the problem statement. Since the task does not mention the parameters α and β, they are both taken to be 1, without loss of generality.

Gamma(3,2) VAE: Where the prior distribution is assumed to be a Gamma distribution with parameters(3,2), as given in the problem statement

Let the prior distribution of the latent space be p and the approximate posterior distribution be q. Our goal is to learn the parameters of q through neural networks. Let z be the vector sampled from the latent space.

Description

In Regular VAEs p is assumed to be a standard normal distribution. To extend VAEs to the case where p is a non standard gaussian, beta distributed or gamma distributed, we face certain unique challenges.

1. Regularization Loss, calculated using Kullback-Liebler(KL) Divergence: The KLD loss changes when the prior changes
2. Method of sampling the latent variable z or modification of the reparameterization trick
3. The parameters of q, the approximate posterior, required to be calculated:

Gaussian(1, 2) VAE

This did work and gave lesser loss than the regular VAE and images generated also seemed better. This can be seen in the Jupyter notebook.

The changes made to the Regular VAE model to enable a Gaussian(1,2) VAE are elaborated.

1. The KL Divergence between two gaussian distributions f and g is given by:

$$
\begin{aligned}
\mathrm{KL}(F||G) &= [\ln(\frac{\sigma_g}{\sigma_f}) + \frac{-\mu_f^2}{2\sigma_f^2} - \frac{-\mu_g^2}{2\sigma_g^2}] + [\frac{2\mu_f}{2\sigma_f^2} - \frac{2\mu_g}{2\sigma_g^2}]\mu_f + [\frac{-1}{2\sigma_f^2} - \frac{-1}{2\sigma_g^2}](\mu_f^2 + \sigma_f^2) \\
&= \ln(\frac{\sigma_g}{\sigma_f}) + \frac{-\mu_f^2}{2\sigma_f^2} + \frac{\mu_g^2}{2\sigma_g^2} + \frac{2\mu_f^2}{2\sigma_f^2} + \frac{-2\mu_g\mu_f}{2\sigma_g^2} + \frac{-\mu_f^2 - \sigma_f^2}{2\sigma_f^2} + \frac{\mu_f^2 + \sigma_f^2}{2\sigma_g^2} \\
&= \ln(\frac{\sigma_g}{\sigma_f}) + \frac{\mu_g^2 - 2\mu_g\mu_f + \mu_f^2 + \sigma_f^2}{2\sigma_g^2} + \frac{-\mu_f^2 + 2\mu_f^2 - \mu_f^2 - \sigma_f^2}{2\sigma_f^2} \\
&= \ln(\frac{\sigma_g}{\sigma_f}) + \frac{(\mu_f - \mu_g)^2 + \sigma_f^2 - \sigma_g^2}{2\sigma_g^2}
\end{aligned}
$$

For a VAE, g is the prior and f is the posterior

For a regular VAE, g has mean =0 and variance = 1 and substituting these values gives the KL loss for a Regular VAE

For a Gaussian(1,2) VAE making a similar substitution for g: mean=1 and variance=2 the KL loss is calculated.

2. Sampling the latent variable z: This remains unchanged since both p and q are gaussian

3. Parameters of q to be calculated: These still remain the mean and variance of q so this remains unchanged

Beta(1, 1) VAE

The changes made to the Regular VAE model to enable a Beta(1,1) VAE are elaborated.

1. The KL Divergence between two beta distributions f and g is given by:

$$
KL(F||G) = ln\Gamma(\alpha f + \beta f)\Gamma(\alpha g)\Gamma(\beta g)\Gamma(\alpha g + \beta g)\Gamma(\alpha f)\Gamma(\beta f) + (\alpha f - \alpha g)(\psi(\alpha f) - \psi(\alpha f + \beta f)) + (\beta f - \beta g)(\psi(\beta f) - \psi(\alpha f + \beta f))
$$

For a VAE, g is the prior and f is the posterior

For a Beta(1,1) VAE making the substitution αg = 1 and βg = 1 the KL loss is calculated in terms of the parameters α and β of the posterior

2. Sampling the latent variable z: Since z is a stochastic variable sampled from a beta distribution, the reparameterization trick is modified.

   $Z = F^{-1}_{\alpha,\beta}[\Phi(\epsilon)]$

   Via the Probability integral transform - Wikipedia

   Additionally this is differentiable, so there shouldn't be problems with calculating the gradient during backpropagation and optimization

   Where F is the cumulative distribution function of the beta distribution and ϵ is the stochastic term ~N(0,1) similar to the Regular VAE

3. Parameters of q to be calculated: Now alpha and beta of the posterior need to be calculated, not the mean and variance as earlier. The current VAE model enabled us already to find the mean and log_variance(these were the parameters being optimized). Using these values, we can use the method of moments estimator for estimating alpha and beta

$$\alpha \text{ hat} = -\frac{\mu(\sigma^2 + \mu^2 - \mu)}{\sigma^2}$$

$$\beta \text{ hat} = \frac{(\sigma^2 + \mu^2 - \mu)(\mu - 1)}{\sigma^2}.$$

Certain problems and analysis:

1. Alpha and beta of the beta distribution, the parameters we are calculating and optimizing are by definition greater than 0. The parameters calculated in the Regular VAE had no such restriction. We need to arrive at positive values of alpha and beta or the model fails.

   How to ensure our alpha and beta values remain > 0?

   Approach tried: By modifying the LeakyReLU to ReLU activation function or to softmax (however softmax leads to a different problem as outlined in 2)

2. Alpha and beta also should not be very close to zero else the distribution *blows up* in a sense.

   How to ensure they remain not very close to zero?

   Softmax's range is 0 to 1 hence it poses a problem

3. The other part of the loss (aside from KL loss) calculated using binary cross entropy loss requires x and x hat to be between 0 and 1 and changing the activation functions as outlined in 1 and 2 caused loss for initial epochs to be nan and infinity, ultimately giving an error regarding input range of BCE loss values

The code ultimately gave a lot of errors regarding using the gamma function for KL loss and conversions between numpy arrays and tensors. It recommended to turn off autograd to perform these operations but turning off autograd in the calculation of the loss function lead to more errors (since backpropagation requires this gradient information) and loss of infinity and nan