M2: Application of Machine Learning

Max Talberg

March 17, 2024

Contents

0.1 Training a Diffusion Model

0.1.1 A regular denoising diffusion model

A diffusion model is comprised of two phases: an encoding (forward diffusion) phase that maps data to a noise distribution and a decoding (reverse diffusion) phase designed to reverse the forward diffusion to generate data, in this instances the MNIST data set.

Encoding (forward process) During the encoding phase the model maps stochastic Gaussian noise to the data over a series of steps, T. The data sample x is transformed into a series of intermediate latent variables $z_1, z_2, ..., z_T$ according to:

$$z_1 = \sqrt{1-\beta_1} \cdot x + \sqrt{\beta_1} \cdot \epsilon_1 \tag{1}$$

$$z_t = \sqrt{1-\beta_t} \cdot z_{t-1} + \sqrt{\beta_t} \cdot \epsilon_t \quad \forall t \in \{2, \ldots, T\} \tag{2}$$

where ϵ_t is noise drawn from a Gaussian distribution and β_t is a hyperparameter that determines the rate at which the noise is integrated into the data. This forward process is a Markov chain as the z_t only depends on the previous step z_{t-1}. This iterative process adds noise onto the sample data x. The diffusion model must understand the encoding phase in order to reverse this process. The diffusion model must understand the encoding phase in order to accurately reverse this process during decoding.

Decoding (reverse process) The decoding phase, learned by the model during training, involves reversing the encoding process to reconstruct the data from noisy latent variables. During training, the model learns to reverse the noise added during the encoding phase recovering the training data. After training the model can generate data from a sample of noise, z_T to z_{T-1} moving through the latent variables until x is recovered.

Training algorithm The training algorithm teaches the model to reverse the encoding process. It simulates the encoding phase by sampling a random timestep t, which corresponds to the noisy data z_t, as well as Gaussian noise ϵ. The aim of the model is to predict the noise ϵ that was added to z_{t-1} to get z_t. This is achieved with a mean squared error (mse) loss function that minimise the difference in the predicted noise and the actual noise ϵ . The loss function is optimised through gradient descent and backpropagation to minimise this loss. The goal is for the model to accurately predict the noise ϵ added at each step, in order to successfully denoise a sample from data.

Algorithm 1 Diffusion model training

Require: Training data x
Ensure: Model parameters ϕ_t

repeat
  for $i \in \mathcal{B}$ **do**
    $t \sim \text{Uniform}[1, \ldots, T]$ ▷ Sample random timestep
    $\epsilon \sim \mathcal{N}(0, I)$ ▷ Sample noise
    $\ell_i = \left\| \sqrt{1-\beta_t} x_i + \sqrt{\beta_t} \epsilon - \phi_t \right\|^2$ ▷ Compute individual loss
  end for
  Accumulate losses for batch and take gradient step
until converged

Sampling algorithm The sampling algorithm utilises this trained model to generate new data from a sample of the noisy latent variable z_T. The model predicts the previous latent variable z_{T-1} by subtracting the predicted noise. The new latent variable receives additional noise from a Gaussian distribution, this is done to simulate the stochastic nature of the forward process improving the quality of samples generated. This process of reversing the encoding is run until the noise is decoded back into data from z_1 to x.

Algorithm 2 Sampling

Require: Model, g_θ, ϕ_t
Ensure: Sample, x

$z_T \sim \text{Norm}_z[0, I]$ ▷ Sample last latent variable
for $t = T, \ldots, 2$ **do**
  $\hat{z}_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\left(z_t - \frac{\beta_t}{\sqrt{1-\alpha_t\beta_t}} g_\theta(z_t, \phi_t)\right)$ ▷ Predict previous latent variable
  $\epsilon \sim \text{Norm}_\epsilon[0, I]$ ▷ Draw new noise vector
  $z_{t-1} = \hat{z}_{t-1} + \sigma_t \epsilon$ ▷ Add noise to previous latent variable
end for
$x = \frac{1}{\sqrt{1-\beta_1}} z_1 - \frac{\sqrt{\beta_1}}{\sqrt{1-\alpha_1\beta_1}} g_\theta(z_1, \phi_1)$ ▷ Generate sample from z_1 without noise

Model The model in this context is a Convolutional Neural Network (CNN) that constitutes the Denoising Diffusion Probabilistic Model (DDPM). The encoding phase is used to generate training data and the decoding phase generates new samples. The model consists of a CNN and DDPM component, the CNN is a 2D model that learns to decode data and the DDPM links this together with the training and sampling algorithms to create a complete diffusion model that generates samples from the MNIST data.

0.1.2 Training a diffusion model

High-level overview of the code The code consists of a CNN and DDPM. The CNN has a building block function that describes the convolutional layer, layer normalisation and an activation function. There is a CNN class that represents the main generative model, this consists of multiple CNN blocks. The DDPM has a scheduler which is where β_t is tuned, this determines the rate of diffusion. The DDPM class encapsulates the diffusion process, including the forward pass for training and the sampling for generating new images.

Training process The diffusion model is trained in Python using PyTorch.

0.1.3 (c)

Exploration, Dimensionality Reduction and Clustering

(a) Density Plot of First 20 Features

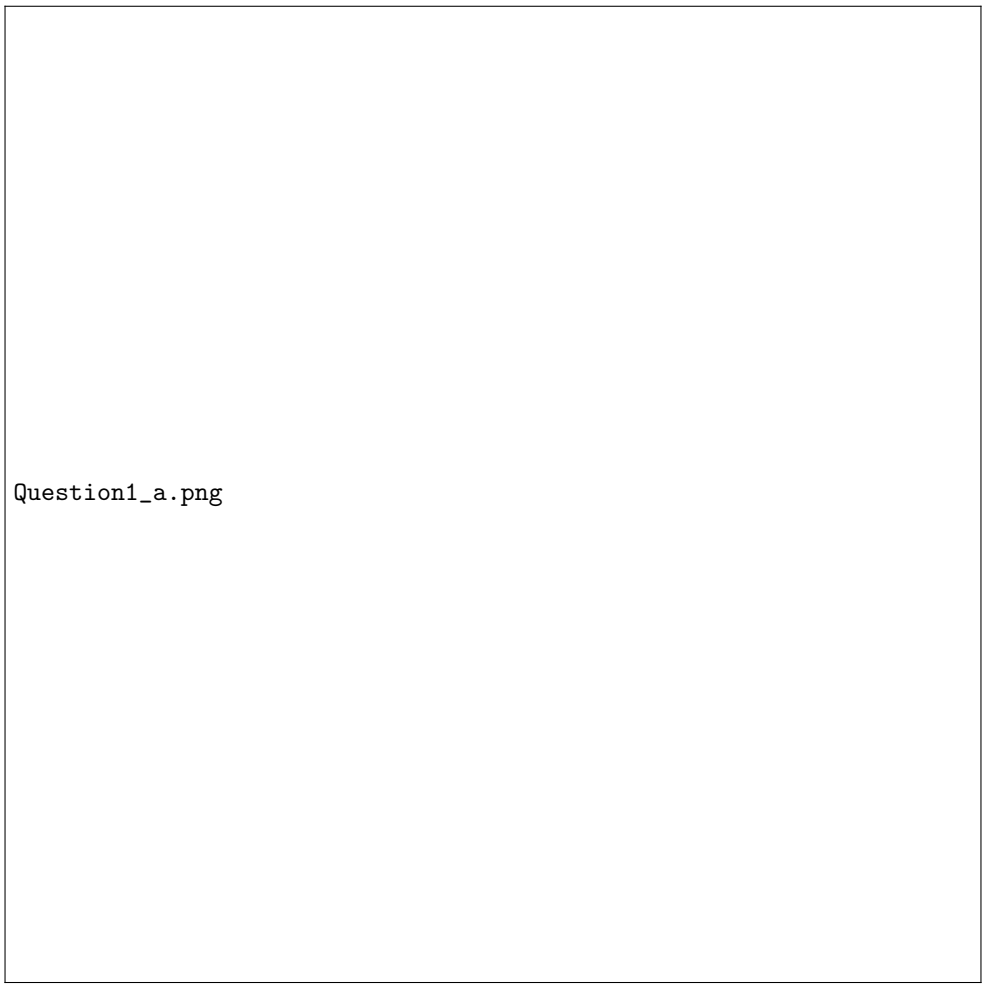

Figure 1: Density plot of the first 20 features, illustrating the distribution of feature values across the dataset.

The density plot in *Figure 1* shows distinct peaks, suggesting potential clusters. The overlapping peaks across several features suggest that some clusters may not be easily separable. The presence of a sharp peak at zero and a broader peak between 2.5 to 5 suggests the potential for at least two clusters.

0.2 Custom Degradation

0.2.1 (a)

0.2.2 (b)

0.2.3 (c)

0.3 Appendix