ECE 8803-GGDL
Homework 4 Solutions
Submitted Monday, November 27

Name: Pratiksha Pai **GT login:** ppai33

1. (a) Attached Jupyter notebook

   (b) i. The parameter λ serves as a crucial scaling factor. It fundamentally controls the rate at which diffusion occurs over time. When $\lambda > 1$, the diffusion accelerates, indicating an increasing spread as time progresses. Conversely, when $\lambda < 1$, the diffusion process slows down, leading to a more gradual spread. If $\lambda = 1$, the rate of diffusion remains constant over time.

   ii. The direction of the score vector in a score-based generative model guides towards areas of greater data density, much like ascending a hill in a landscape. The magnitude of the score, on the other hand, indicates the distance from the mode of the distribution – the greater the distance, the larger the magnitude.
   In the case of a multi-modal distribution, such as a Gaussian Mixture Model (GMM), the overall score at any point is effectively a weighted average of the scores from the individual modes. These weights are proportional to the density each mode contributes at that particular point. Therefore, the overall score at any given point is predominantly influenced by the modes that are 'closest' or most relevant to that point.

   iii. The reverse diffusion samples (in blue) closely match the contour patterns of the samples from the original Gaussian Mixture Model (GMM) (in orange). This suggests that the reverse diffusion process is accurately capturing the original distribution, reproducing the modes and variances of the GMM effectively

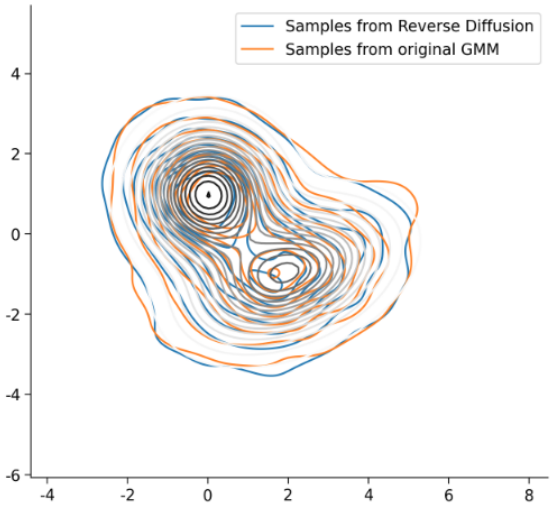

Figure 1: GMM Reverse.

(c) The score in multi-modal distributions guides 'denoising' by directing samples towards higher probability regions. The denoising objective loss, $\|\sigma_t s_\theta(x+\sigma_t z, t)+z\|^2$, minimizes when the model's predicted score s_θ accurately estimates the true score, allowing for effective reversal of the diffusion noise. Ideally, the loss could reach 0 if the model perfectly learns the score function, denoising samples precisely. However, practically, due to model and optimization limitations, the objective is to approximate this ideal as closely as possible.

(d) i. Time modulation is implemented by adding the output of t_mod layers to the upsampled features, for example, $h = self.tconv3(torch.cat([h, h3], dim = 1)) + self.t\_mod6(embed)$.

The final normalization, where the score target is divided by the noise scale σ_t, has pros and cons:

- **Pro**: It normalizes the target score $s(x,t)$ to z/σ_t, which means the neural network primarily models the target z that maintains a consistent variance of approximately 1 across different times t.
- **Con**: Near $t \approx 0$, this normalization amplifies both noise and signal significantly, potentially leading to large errors under low noise conditions. The loss weighting function mitigates this by reducing the influence of periods with small σ_t in the overall loss.

ii. Yes, the generated data seems to be noisy.

2. (a) When irreps have partially or completely redundant entries, it affects the basis they constitute by making it overcomplete. For instance, consider the group of planar rotations $SO(2)$. The irreducible representations (irreps) of this group are two-dimensional, but they contain redundant information. As such, each pair of columns in the matrix repre-

sentation spans the same space. This redundancy implies that the full set of functions over the group can be represented using fewer irreps than the dimension of the space, leading to an overcomplete basis.

Consequently, the regular representation, which is built from the direct sum of all irreps, each weighted by its dimension, will have each irrep appearing only once instead of its dimensionality times. This makes the regular representation more efficient, as it avoids unnecessary repetition of the same information.

(b)
- Transforming an image with the group $G = C_4$ does not need interpolation. This group represents rotations by 0, 90, 180, and 270 degrees—angles that align with the pixel grid of a digital image. When an image is rotated by angles in the C_4 group, each pixel in the rotated image corresponds directly to a pixel in the original image. Interpolation is needed when transformations involve angles or scaling factors.
- The group SO(2) includes all planar rotations, but for pixel-based digital images, only rotations by $\frac{\pi}{2}$ increments align with the grid, avoiding interpolation. Other angles introduce noise due to interpolation, affecting the perfect equivariance of rotational transformations. This constraint also applies to the rotation of convolutional filters in image processing, where discrete grid representation leads to interpolation and subsequent noise.

(c)
- No trained model achieves perfect invariance to $SO(2)$ due to the pixel grid discretization requiring interpolation for non-$\pi/2$ rotations, introducing noise and distortion. The $SO(2)$-equivariant model, however, demonstrates greater stability across various rotation angles, as shown by its more consistent accuracy on the rotated test set, compared to the C_4-equivariant model.
- Perfect equivariance to rotations by multiples of $\pi/2$ is exhibited by both models. This is because such rotations align with the pixel grid, eliminating the need for interpolation and preserving image integrity, thus maintaining perfect equivariance for these specific rotations.

3. Let P be a permutation matrix that represents a permutation of the nodes of a graph. For a set of node features X and an adjacency matrix A, we define $X' = PX$ and $A' = PAP^T$ to be the permuted node features and adjacency matrix, respectively.

   The goal is to show that if $F(X, A)$ is permutation equivariant, then for the function F defined by the permutation invariant function ϕ, it follows that $F(X', A') = PF(X, A)$.

   Proof:

   Consider the function $F(X, A)$ applied to the permuted inputs X' and A':

$$F(X', A') = \begin{bmatrix} \phi(X'_1, X'_{N'_1}) \\ \phi(X'_2, X'_{N'_2}) \\ \vdots \\ \phi(X'_n, X'_{N'_n}) \end{bmatrix}.$$

Here, X'_i is the feature vector of the i-th node after permutation, and $X'_{N'_i}$ represents the features of the neighboring nodes after permutation.

Due to the permutation invariance of ϕ, for each node i, we have:

$$\phi(X'_i, X'_{N'_i}) = \phi(PX_i, PX_{N_i}),$$

where PX_{N_i} are the permuted neighborhood features of X_i.

Since ϕ is invariant to the ordering of its second argument, the above can be rewritten as:

$$\phi(PX_i, PX_{N_i}) = P\phi(X_i, X_{N_i}).$$

Applying P to $F(X, A)$ rearranges its rows in the same way that X was permuted to obtain X', resulting in:

$$PF(X, A) = P \begin{bmatrix} \phi(X_1, X_{N_1}) \\ \phi(X_2, X_{N_2}) \\ \vdots \\ \phi(X_n, X_{N_n}) \end{bmatrix} = \begin{bmatrix} \phi(PX_1, PX_{N_1}) \\ \phi(PX_2, PX_{N_2}) \\ \vdots \\ \phi(PX_n, PX_{N_n}) \end{bmatrix}.$$

Therefore, each row i of $F(X', A')$ matches row $\pi(i)$ of $PF(X, A)$, and we can conclude:

$$F(X', A') = PF(X, A).$$

This completes the proof that F is permutation equivariant, as the application of P to $F(X, A)$ produces the same result as applying F to the permuted graph characterized by X' and A'.

4. **Graph Convolutional Network (GCN)**

- **Edge Function** (ψ): Typically undefined or an identity function.
- **Node Function** (ϕ):
$$\phi(x_v) = \text{ReLU}(W \cdot x_v + b)$$
Where W and b are the weight matrix and bias vector, respectively.
- **Global Aggregation** (ρ):
$$\rho(\{h_{uv} | u \in N_v\}) = \text{mean}(\{h_{uv} | u \in N_v\})$$

Graph Attention Network (GAT)

- **Edge Function** (ψ): Computes attention coefficients:
$$\psi(x_{uv}) = \text{softmax}_u(e_{uv})$$
Where e_{uv} is the attention score.

- **Node Function** (ϕ): Similar to GCN's ϕ.
- **Global Aggregation** (ρ): Weighted sum using attention coefficients:

$$\rho(\{h_{uv}|u \in N_v\}) = \sum_{u \in N_v} \alpha_{uv} h_{uv}$$

  Where α_{uv} are normalized attention scores from ψ.

Message Passing Neural Network (MPNN)

- **Edge Function** (ψ): Computes a message:

$$\psi(x_{uv}, x_u, x_v) = M(x_u, x_{uv}, x_v)$$

  Where M is the message function.

- **Node Function** (ϕ): Updates node features:

$$\phi(h_v, m_{N_v}) = U(h_v, m_{N_v})$$

  Where U is the update function.

- **Global Aggregation** (ρ): Aggregates messages:

$$\rho(\{m_{uv}|u \in N_v\}) = \sum_{u \in N_v} m_{uv}$$