Séance 2 : Diffusion sur Données Numériques Simples

Julien Perez

December 12, 2024

Introduction à la Séance 2

Objectifs de la Séance :

- Appliquer les modèles de diffusion à des données numériques simples.
- Analyser les trajectoires de diffusion et comprendre le processus de génération.
- Mettre en pratique les concepts théoriques avec des exercices sur des distributions numériques.
- Maîtriser l'utilisation de la bibliothèque `Diffuser`.

Modèles de Diffusion pour Distributions Simples

Exemple : Distribution Gaussienne Multimodale

- Une distribution gaussienne multimodale a plusieurs pics (modes).
- Objectif : Apprendre à modéliser et générer de telles distributions.

Pourquoi les Modèles de Diffusion ?

- Capacité à capturer des structures complexes dans des distributions simples.

Rappel : Processus de Diffusion

Processus de Bruitage :

- Le bruit est ajouté de manière progressive pour rendre les données aléatoires.
- Chaque étape est une transformation linéaire avec de la variance ajoutée.

Processus de Débruitage :

- Apprendre à inverser le bruitage pour retrouver la structure originale.

Pourquoi Modéliser avec la Diffusion ?

Modélisation des Distributions Complexes :

- Les distributions simples (ex : gaussiennes univariées) ne capturent pas les structures multimodales présentes dans des données réelles.
- Les modèles de diffusion sont efficaces pour modéliser des distributions complexes, même avec des modes multiples.

Comparaison :

- Modèles traditionnels (ex : VAE) vs. Modèles de diffusion : une meilleure flexibilité de génération.

Cas d'Utilisation :

- Génération de données synthétiques réalistes.
- Applications dans la reconstruction et la super-résolution.

Le Concept de Multimodalité

Définition :

- Une distribution est dite multimodale lorsqu'elle a plusieurs pics distincts.

Exemple :

- Considérez une combinaison de deux gaussiennes : $\mathcal{N}(\mu_1, \sigma_1^2)$ et $\mathcal{N}(\mu_2, \sigma_2^2)$.
- Si $\mu_1 \neq \mu_2$, la distribution résultante présente deux modes.

Problèmes Posés :

- Modélisation difficile avec des approches classiques.
- Le processus de diffusion permet de lisser et de reconstruire de telles distributions.

Distribution Gaussienne Multimodale

Définition :

- Une distribution avec plusieurs pics ou modes.
- Exemple : Combinaison de plusieurs gaussiennes univariées avec des moyennes et des variances différentes.

Propriétés :

- Difficulté accrue de modélisation par des modèles simples.

Application aux Données Numériques Simples

Étapes :

- Étape 1 : Initialisation de la distribution cible (ex : gaussienne multimodale).
- Étape 2 : Application du processus de diffusion pour ajouter du bruit.
- Étape 3 : Apprentissage du modèle de débruitage pour reconstruire les données.

Résultats Attendues : Reconstruction efficace de la distribution originale.

Analyse des Trajectoires de Diffusion

Visualisation des Trajectoires :

- Observer comment les points de données se déplacent sous l'effet du bruitage.
- Étudier les variations de la densité de probabilité au cours du temps.

Intuition :

- Comprendre comment le modèle apprend les relations entre les données bruitées et non bruitées.

Le Processus de Bruitage en Détail

Étapes Progressives :

- À chaque pas de temps t, un bruit gaussien est ajouté : $x_t = \sqrt{1-\beta_t} x_{t-1} + \sqrt{\beta_t}\epsilon$, où $\epsilon \sim \mathcal{N}(0, I)$.
- Les paramètres β_t contrôlent la quantité de bruit ajoutée.

Objectif :

- Transformer les données en une distribution de bruit blanc après plusieurs étapes.

Visualisation :

- Chaque étape montre une dégradation progressive de la structure originale des données.

Reconstruction des Données par Débruitage

Processus Inverse :

- Utilisation du modèle pour générer des données à partir de bruit aléatoire.
- Importance d'apprendre une distribution réaliste des données.

Applications Pratiques :

- Génération de nouvelles données réalistes à partir de distributions bruitées.

Étude de la Distribution Finale

Distribution Complètement Bruitée :

- Après de nombreuses étapes de bruitage, la distribution des données devient une distribution gaussienne isotrope.
- Cela facilite la modélisation car la distribution finale est connue.

Rôle dans la Génération :

- La distribution bruitée sert de point de départ pour le processus de débruitage.

Importance Théorique :

- Permet de lier la théorie des processus de Markov à la génération de données.

Processus de Débruitage Détaillé

Apprentissage du Processus Inverse :

- Le modèle apprend à estimer $p_\theta(x_{t-1}|x_t)$, une distribution gaussienne paramétrée.

Paramètres :

- Moyenne $\mu_\theta(x_t, t)$: Prédit la position non bruitée.
- Variance $\Sigma_\theta(x_t, t)$: Représente l'incertitude.

Pourquoi est-ce Complexe ?

- Les prédictions doivent être précises à chaque étape pour une reconstruction réaliste.
- Utilisation de techniques avancées comme les réseaux neuronaux U-Net.

Diffusion et Optimisation de l'Objectif

Objectif d'Apprentissage :

- Minimiser $\mathcal{L} = \mathbb{E}_{x_0,\epsilon,t}[\|\epsilon - \epsilon_\theta(x_t, t)\|^2]$.
- Où ϵ_θ est le réseau de prédiction du bruit.

Pourquoi cette Formule ?

- Approche basée sur la reconstruction du bruit au lieu de la donnée elle-même.
- Permet une meilleure capture des variations de la distribution.

Conséquences :

- Une optimisation efficace requiert des données en grande quantité.

Méthodes d'Accélération du Débruitage

Problème :

- Le processus de débruitage est lent en raison des nombreuses étapes.

Solutions :

- **DDIM (Denoising Diffusion Implicit Models) :** Réduction du nombre d'étapes sans compromettre la qualité.
- **Guidance Conditionnelle :** Diriger le processus pour des générations plus rapides et contrôlées.

Impact :

- Réduit le temps de calcul nécessaire pour des applications pratiques.

Exemple Numérique : Trajectoire de Diffusion

Exemple :

- Considérons une distribution initiale de données : points dans un espace unidimensionnel.
- Visualisation de la trajectoire de ces points lorsqu'ils subissent le bruitage.

Analyse :

- Chaque étape de diffusion dilue la structure originale des points.

Exercice : Génération avec Guidance

Objectif :

- Implémenter un modèle de diffusion simple avec une guidance conditionnelle.

Étapes :

1. Créez une distribution cible avec plusieurs modes.
2. Ajoutez du bruit et essayez de guider la génération vers un mode spécifique.
3. Comparez les résultats avec et sans guidance.

Questions de Réflexion :

- Comment la guidance améliore-t-elle la qualité de génération ?
- Quels sont les compromis en termes de temps de calcul ?

Mise en Pratique : Exercices

Exercice 1 : Génération de Données

- Créez une distribution gaussienne multimodale.
- Appliquez le processus de diffusion et observez les changements.

Exercice 2 : Reconstruction de Données

- Utilisez un modèle simple pour débruiter et reconstruire les données.
- Analysez la qualité de la reconstruction.

Code: Modèle de Diffusion

Définition du Modèle de Diffusion :

```
import torch
import torch.nn as nn
import torch.optim as optim

class DiffusionModel(nn.Module):
    def __init__(self, input_dim, hidden_dim):
        super(DiffusionModel, self).__init__()
        # Définition des couches linéaires
        self.fc1 = nn.Linear(input_dim + 1, hidden_dim)  # +1 pour le temps t
        self.fc2 = nn.Linear(hidden_dim, input_dim)

    def forward(self, x, t):
        # Ajout d'informations temporelles t
        t = t.unsqueeze(1)  # Ajouter une dimension pour concaténer avec x
        x_t = torch.cat([x, t], dim=-1)
        h = torch.relu(self.fc1(x_t))  # Application de la fonction d'activation Re
        return self.fc2(h)
```

Code: Modèle de Diffusion

Processus d'Entraînement :

```
# Initialisation du modèle et de l'optimiseur
input_dim = 100
hidden_dim = 128
diffusion_model = DiffusionModel(input_dim=input_dim, hidden_dim=hidden_dim)
optimizer = optim.Adam(diffusion_model.parameters(), lr=1e-4)

# Fonction de perte pour le modèle de diffusion
def diffusion_loss(x, x_noisy, noise_pred):
    return nn.MSELoss()(noise_pred, x_noisy - x)
```

Code: Modèle de Diffusion

Processus d'Entraînement :

```
# Paramètres d'entraînement
epochs = 100
T = 1000  # Nombre total de pas de temps

# Boucle d'entraînement
for epoch in range(epochs):
    for x in dataloader:
        t = torch.randint(0, T, (x.size(0),))  # Temps aléatoire
        noise = torch.randn_like(x)
        x_noisy = x + noise * torch.sqrt(t.float() / T)  # Ajout de bruit

        optimizer.zero_grad()  # Réinitialisation des gradients
        noise_pred = diffusion_model(x, t.float())  # Prédiction du bruit
        loss = diffusion_loss(x, x_noisy, noise_pred)  # Calcul de la perte
        loss.backward()  # Rétropropagation
        optimizer.step()  # Mise à jour des poids
```

Exercice 2: Apprentissage de Trajectoires en 2D I

Objectif : Utiliser le modèle de diffusion pour apprendre des trajectoires échantillonnées dans un espace 2D.

Énoncé :

- Générez un jeu de données de trajectoires 2D. Chaque trajectoire est une séquence de points (x, y) dans un espace 2D.
- Adaptez le modèle de diffusion pour traiter des séquences de points 2D.
- Entraînez le modèle sur le jeu de données généré.
- Évaluez la performance du modèle en générant de nouvelles trajectoires et en les comparant aux trajectoires originales.

Exercice 2: Apprentissage de Trajectoires en 2D II

Indications :

- Utilisez des fonctions de génération de trajectoires aléatoires ou des trajectoires suivant une certaine distribution.
- Adaptez la dimension d'entrée du modèle pour accepter des séquences de points 2D.
- Visualisez les trajectoires générées pour évaluer la performance.

Introduction aux modèles de diffusion

Les modèles de diffusion sont une classe de modèles génératifs utilisés pour :

- La génération d'images
- La génération audio
- D'autres tâches créatives et de manipulation avancées

Bibliothèque Diffusers

La bibliothèque Diffusers offre :

- Des implémentations de modèles de diffusion état de l'art
- Une API simple pour la génération d'images et d'audio
- Des outils pour l'entraînement et le fine-tuning de modèles

Fonctionnalités principales

- Génération d'images
- Génération audio
- Inpainting et outpainting
- Super-résolution
- Conversion texte-image

Installation

```
pip install diffusers transformers
```

Exemple de génération d'image

```
pipe = StableDiffusionPipeline.from_pretrained("runwayml/stable-diffusion-v1-5",
torch_dtype=torch.float16 ) pipe = pipe.to("cuda")
prompt = "a photo of an astronaut riding a horse on mars" image =
pipe(prompt).images image.save("astronaut_rides_horse.png")
```

Modèles populaires

- Stable Diffusion
- DALL-E 2
- Midjourney (non disponible via Hugging Face)
- Imagen

Facilité d'utilisation

- API intuitive
- Documentation complète
- Exemples et tutoriels disponibles

Flexibilité

- Support de multiples frameworks (PyTorch, TensorFlow, JAX)
- Possibilité de personnaliser les pipelines
- Intégration facile dans des projets existants

Communauté et ressources

- Large communauté d'utilisateurs et de contributeurs
- Nombreux modèles pré-entraînés disponibles
- Forums et canaux de support actifs

Domaines d'application

- Art et design
- Publicité et marketing
- Jeux vidéo et cinéma
- Recherche scientifique

Cas d'utilisation

- Création de visuels pour les réseaux sociaux
- Prototypage rapide en design
- Génération de textures pour les jeux vidéo
- Restauration d'images anciennes

Tendances et développements

- Amélioration de la qualité et de la résolution des générations
- Intégration de nouvelles modalités (vidéo, 3D)
- Optimisation pour des appareils à faible puissance

Défis et considérations éthiques

- Biais dans les données d'entraînement
- Droits d'auteur et propriété intellectuelle
- Utilisation malveillante (deepfakes, désinformation)

Génération d'image à partir de texte

```
from diffusers import StableDiffusionPipeline
import torch

pipe = StableDiffusionPipeline.from_pretrained("runwayml/stable-diffusion-v1-5")
pipe = pipe.to("cuda")

prompt = "un chat portant un chapeau de cowboy"
image = pipe(prompt).images
image.save("chat_cowboy.png")
```

Inpainting (retouche d'image)

```
from diffusers import StableDiffusionInpaintPipeline
from PIL import Image

pipe = StableDiffusionInpaintPipeline.from_pretrained(
"runwayml/stable-diffusion-inpainting"
)
pipe = pipe.to("cuda")
image = Image.open("image_originale.png")
mask_image = Image.open("masque.png")
prompt = "un arbre fleuri"

result = pipe(prompt=prompt, image=image, mask_image=mask_image).images
result.save("image_retouchee.png")
```

Super-résolution

```
from diffusers import StableDiffusionUpscalePipeline
import torch
from PIL import Image

model_id = "stabilityai/stable-diffusion-x4-upscaler"
pipeline = StableDiffusionUpscalePipeline.from_pretrained(
model_id, torch_dtype=torch.float16)

pipeline = pipeline.to("cuda")
image = Image.open("image_basse_resolution.png")
prompt = "une image haute résolution d'un paysage"

upscaled_image = pipeline(prompt=prompt, image=image).images
upscaled_image.save("image_haute_resolution.png")
```

Génération d'image guidée par croquis

```
from diffusers import StableDiffusionControlNetPipeline, ControlNetModel
from diffusers.utils import load_image

controlnet = ControlNetModel.from_pretrained(
"lllyasviel/sd-controlnet-scribble")
pipe = StableDiffusionControlNetPipeline.from_pretrained(
"runwayml/stable-diffusion-v1-5", controlnet=controlnet)
pipe = pipe.to("cuda")

croquis = load_image("croquis.png")
prompt = "un paysage coloré basé sur ce croquis"
image = pipe(prompt, image=croquis).images
image.save("paysage_genere.png")
```

Génération de texte à partir d'image

```
from transformers import BlipProcessor, BlipForConditionalGeneration
from PIL import Image

processor = BlipProcessor.from_pretrained("Salesforce/blip-image-captioning-base")
model = BlipForConditionalGeneration.from_pretrained("Salesforce/blip-image-captioning-base")

image = Image.open("image.jpg")
inputs = processor(images=image, return_tensors="pt")
output = model.generate(**inputs)
caption = processor.decode(output, skip_special_tokens=True)
print(caption)
```

Génération audio

```
from diffusers import AudioLDMPipeline
import torch
import scipy

pipe = AudioLDMPipeline.from_pretrained("cvssp/audioldm",
                                        torch_dtype=torch.float16)
pipe = pipe.to("cuda")
prompt = "Le son d'une forêt tropicale avec des oiseaux qui chantent"
audio = pipe(prompt, num_inference_steps=10, audio_length_in_s=5.0).audios
scipy.io.wavfile.write("audio_genere.wav", rate=16000, data=audio)
```

Conclusion de la Séance 2

Résumé :

- Application des modèles de diffusion à des données numériques simples.
- Analyse des trajectoires de diffusion et importance de la reconstruction.
- Mise en pratique pour renforcer la compréhension des concepts.

À Suivre : Approfondissement sur des données plus complexes et l'application pratique en machine learning.