TP5. Auto-encodeurs.

Frédéric Richard

Cours apprentissage statistique et réseaux de neurones
Master mathématiques appliquées, statistique,
Parcours Data Science.

2025, AMU

L'objectif de ce TP est d'apprendre à construire et à utiliser les auto-encodeurs. On travaillera sur la base MNIST. Dans un premier temps, on comparera des auto-encodeurs denses et convolutionnels. Puis, on appliquera les auto-encodeurs pour le débruitage d'images.

1 Préambule.

Déclaration des librairies et méthodes utiles au TP:

[3]:
```
from torch.nn import Module

# A compléter.
```

Préparation des données.

Charger la base de données MNIST et préparer-la pour le traitement en pytorch en séparant les données d'apprentissage et de test.

2 1. Auto-encodeurs denses.

On va définir plusieurs auto-encodeurs à couches denses en utilisant la classe **Module** de la librairie *torch.nn*.

1.1. Modèle Dense n°1 Il s'agit d'un autoencodeur dont l'encodeur et le décodeur ne comporte qu'une seule couche dense. La couche cachée de l'auto-encodeur est de taille K. L'activation à la suite des couches denses est au choix. La taille de l'image en entrée, le paramètre K et l'activation seront entrées en argument de la fonction qui définit l'auto-encodeur (voir ci-dessous).

Partie	**Couche**	**Spécifications**	**Dimension de la sortie**	**Nombre de paramètres**
Encodeur	Input	-	(M, N)	-
	Vectorisation	-	.....	-
	Dense 1	K cellules, activation	.....	.....

Partie	Couche	Spécifications	Dimension de la sortie	Nombre de paramètres
Décodeur	Dense 2	M x N cellules, activation sigmoïd	.....	.....
	Remise en forme de l'image	-	(M, N)	-

On pourra définir le modèle en remplissant le programme suivant:

[1]:
```
# Définition de l'encodeur.
class Encoder(Module):
    def __init__(self, M, N, K, activation):
        super(Encoder, self).__init__()
        # Initialisation des paramètres.
        self.M = M
        self.N = N
        self.K = K
        self.activation = activation
        # Définition des couches.
        # A compléter.

    def forward(self, x):
        x.view(x.size(0), -1)  # Vectorisation des images.
        # A compléter.
        return x

# Définition du décodeur.
class Decoder(Module):
    def __init__(self, M, N, K, activation):
        super(Decoder, self).__init__()
        # Initialisation des paramètres.
        self.M = M
        self.N = N
        self.K = K
        self.activation = activation
        # Définition des couches.
        # A compléter.

    def forward(self, x):
        # A compléter
        x = x.view(x.size(0), 1, self.M, self.N)  # Remise en forme de l'image.
        return x

# Définition de l'auto-encodeur.
class AutoEncoder(Module):
```

```
    def __init__(self, im_shape, K, activation=None):
        super(AutoEncoder, self).__init__()
        # Initialisation des paramètres.
        self.M = im_shape[0]
        self.N = im_shape[1]
        self.K = K
        # Définition de l'encodeur et du décodeur.
        self.encoder = Encoder(M, N, K, activation)
        self.decoder = Decoder(M, N, K, activation)

    def forward(self, x):
        return self.decodeur(self.encoder(x))
```

1.2. Modèle Dense n°2. On définira ensuite un auto-encodeur à plusieurs couches denses.

Partie	**Couche**	**Spécifications**	**Dimension de la sortie**	**Nombre de paramètres**
	Input	-	(M, N)	-
Encodeur	Vectorisation	-	.....	-
	Dense 1	128 cellules	.....	.....
	Activation	au choix	-	-
	Dense 2	K cellules	.....	.....
	Activation	au choix	-	-
Décodeur	Dense 3	128 cellules	.....	.....
	Activation	au choix	-	-
	Dense 4	M x N cellules	.....	.....
	Activation	sigmoïd	-	-
	Remise en forme de l'image	-	(M, N)	-

Exercice 1

1. Instancier
   - le modèle Dense_1 sans activation (None) en prenant une couche cachée de taille $K = 64$.
   - le modèle Dense_1 avec une activation relu en prenant une couche cachée de taille $K = 64$.
   - le modèle Dense_2 avec une activation relu en prenant une couche cachée de taille $K = 64$.
2. Compléter les tableaux de description des modèles ainsi instanciés.
3. Comparer la complexité des modèles (en termes de nombre de paramètres).
4. Faire l'apprentissage des modèles.
5. Comparer leurs performances sur la base de test.
6. Visualiser quelques résultats en comparant l'image initiale et l'image reconstruite par auto-encodeur.

2.1 2. Auto-encodeurs convolutionnels.

On s'intéresse maintenant à un auto-encodeur à couches de convolution défini comme suit.

Partie	**Couche**	**Spécifications**	**Dimension de la sortie**	**Nombre de paramètres**
	Input	-	(M, N)	-
Encodeur	Convolution 2D n°1	16 filtres de taille 3 x 3, stride=2, padding=1	.....	.....
	Convolution 2D n°1	16 filtres de taille 3 x 3, stride=2, padding=1	.....	.....
	Activation	relu	-	-
	Convolution 2D n°2	32 filtres de taille 3 x 3, stride=2, padding=1	.....	.....
	Activation	relu	-	-
	Convolution 2D n°3	64 filtres de taille 7 x 7, stride=1, padding=0	.....	.....
Décodeur	Convolution transposée 2D n°1	64 filtres de taille 7 x 7, stride=1, padding=0	.....	.....
	Activation	relu	-	-
	Convolution transposée 2D n°2	32 filtres de taille 3 x 3, stride=2, padding=1, output_padding=1	.....	.....
	Activation	relu	-	-
	Convolution transposée 2D n°3	16 filtres de taille 3 x 3, stride=2, padding=1, output_padding=1	.....	.....
	Activation	sigmoïd	-	-

Exercice 2

1. Instancier le modèle.
2. Compléter le tableau de description du modèle.
3. Calculer la complexité du modèle et la comparer avec celles des modèles à couches denses.
4. Comparer la taille de la couche cachée du modèle avec celles des modèles à couches denses.
5. Faire l'apprentissage du modèle.
6. Comparer ses performances avec celles des modèles à couches denses sur la base de test.
7. Visualiser quelques résultats.

2.2 2. Application au débruitage d'images.

Les auto-encodeurs peuvent être utilisés pour débruiter les images. Pour cela, on fait l'apprentissage de l'auto-encodeur en l'appliquant à des images bruitées et faisant en sorte de rapprocher leurs images de sortie des images non bruitées. Le modèle apprend ainsi à produire une image débruitée à partir de l'image bruitée. Son action devient alors celle d'un filtre.

Exercice 3

1. Créer une base d'exemples (x_i, y_i) où y_i est une image de la base MNIST et $x_i = y_i + n_i$ où n_i est un bruit additif gaussien d'espérance 0 et de variance fixée.
2. Faire l'apprentissage de l'auto-encodeur sur cette base d'exemples en prenant x_i en entrée et minimisant le critère des moindres carrés entre la sortie de l'auto-encodeur et l'image y_i.
3. Evaluer la similarité entre les images sans bruit et les images débruités sur une base de test.
4. Visualiser l'effet du filtre quelques images bruitées.