INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Réseaux de neurones

Transposer comme un dieu, différentier comme une déesse: l'intelligence non artificielle des maths.

Poly de TDs

3MIC semestre 7

Dr. Frédéric de Gournay

Publié par INSA de Toulouse

frederic@degournay.fr

Première édition, Janvier 2024.

Cahier de TDs

1 Calculer des adjoints

(15 mins) Exercice 1.1

L'objectif est de retrouver la formule de l'adjoint dans le cas où le produit scalaire n'est pas Euclidien, ou dans le cas où le base n'est pas orthonormée. On se donne E et F deux espaces vectoriels réels de dimension respectives n et m finies. On se donne $(e_i)_{i=1,\dots n}$ et $(f_j)_{j=1,\dots,m}$ deux bases de E et F. On se donne deux produits scalaires sur E et F notés $\langle \bullet, \bullet \rangle_E$ et $\langle \bullet, \bullet \rangle_F$. Pour tout vecteur $u \in E$ (resp. $v \in F$), on note $\tilde{u} \in \mathbb{R}^n$ (resp. $\tilde{v} \in \mathbb{R}^n$) le vecteur des coordonnées de u (resp. de v), i.e. le vecteur tel que

$$u = \sum_{i=1}^{n} \tilde{u}[i] e_i \quad (resp.\ v = \sum_{j=1}^{m} \tilde{v}[j] f_j).$$

Finalement, on note $(\bullet, \bullet)_n$ le produit scalaire usuel de $\mathbb{R}^n$.

- Soit S_E la matrice de taille $n \times n$ donnée par $S_E[i,j] = \langle e_i, e_j \rangle_E$ (on définira de manière similaire S_F plus tard).
  1. Montrer rapidement que la base (e_i) est orthonormale si et seulement si S_E vaut l'identité.
  2. Montrer que pour tout $a, b \in E$, on a $\langle a, b \rangle_E = \langle \tilde{a}, S_E \tilde{b} \rangle_n$
  3. Montrer rapidement que S_E est une matrice symétrique définie positive.
- Pour tout opérateur $\mathcal{A}$ de E dans F, on note $\tilde{\mathcal{A}}$ la matrice $\mathcal{M}_{mn}(\mathbb{R})$ tel que pour tout u de E :

  $$\widetilde{(\mathcal{A}u)} = \tilde{\mathcal{A}}\tilde{u},$$

  ou encore, pour tout i, $\mathcal{A}e_i = \sum_{j=1}^{m} \tilde{\mathcal{A}}[j,i] f_j$. Montrer que $\widetilde{(\mathcal{A}^\star)} = S_E^{-1} \tilde{\mathcal{A}}^T S_F$

(15 mins) Exercice 1.2

Soit $m, n, p \in \mathbb{N}$. Pour tout $x \in \mathcal{M}_{np}(\mathbb{R})$ et $A \in \mathcal{M}_{mn}(\mathbb{R})$, on nomme $A \cdot x \in \mathcal{M}_{mp}(\mathbb{R})$ la matrice définie par, pour tout $1 \leq i \leq m$ et $1 \leq j \leq p$.

$$(A \cdot x)_{i,j} = \sum_{k=1}^{n} A_{i,k} x_{k,j}.$$

Donner les adjoints des opérateurs $A \mapsto A \cdot x$ et $x \mapsto A \cdot x$, si on considère les produits scalaires usuels sur les matrices. On concluera que leurs adjoints sont, respectivement, $B \mapsto B \cdot x^T$ et $B \mapsto A^T \cdot B$.

(30 mins) Exercice 1.3

L'objectif est de construire des espaces de discrétisation de l'ensemble des fonctions.

- On se donne $n+2$ points $a = x_0 < x_1 < \cdots < x_n < x_{n+1} = b$. Pour tout i entre 0 et $n+1$, on note $x_{i+\frac{1}{2}}$ le milieu du segment $[x_i, x_{i+1}]$, on a donc

  $$x_{i+1/2} = \frac{x_i + x_{i+1}}{2},$$

- Soit $C_c^\infty([a,b])$ l'ensemble des fonctions C^∞ sur $[a,b]$ qui s'annulent sur un voisinage de a et de b. On note E l'espace vectoriel des $u \in \mathbb{R}^{n+2}$ tels que $u_0 = u_{n+1} = 0$ et F l'espace vectoriel $\mathbb{R}^{n+1}$. pour les diférentier les vecteurs

$u \in E$ seront notés $(u_i)_{0 \le i \le n+1}$ et les vecteurs $v \in F$ seront notés $(v_{i+\frac{1}{2}})_{0 \le i \le n}$ On construit les applications de discrétisation π_E et π_F qui vont de $C_c^\infty([a,b])$ vers respectivement E et F et qui sont définies par

$$\pi_E(\phi)_i = \phi(x_i) \quad \text{ et } \pi_F(\phi)_{i+\frac{1}{2}} = \phi(x_{i+\frac{1}{2}}).$$

En d'autres termes, l'espace E est la discrétisation des fonctions $C_c^\infty([a,b])$ prises aux points x_i et l'espace F est la discrétisation des fonctions $C_c^\infty([a,b])$ prises aux points $x_{i+\frac{1}{2}}$.

- On se donne sur E et F des produits scalaires qui sont des versions discrétisées de $\langle \phi, \psi \rangle = \int \phi\psi$.

$$\langle u, v \rangle_E = \sum_{i=0}^{n} (x_{i+1} - x_i) \frac{u_{i+1}v_{i+1} + u_i v_i}{2} \text{ et } \langle u, v \rangle_F = \sum_{i=0}^{n} (x_{i+1} - x_i) u_{i+\frac{1}{2}} v_{i+\frac{1}{2}}.$$

Autrement dit, pour E on regarde une approximation par la méthode des trapèzes et pour F par le point du milieu.

- Si $\mathcal{D}$ est l'endomorphisme de $C_c^\infty([a,b])$ défini par $\mathcal{D} : \phi \mapsto \phi'$, une discrétisation logique de $\mathcal{D}$ est l'opérateur $D : E \to F$ défini par

$$\forall u \in E \quad (Du)_{i+1/2} = \frac{u_{i+1} - u_i}{x_{i+1} - x_i}$$

1. Vérifier que

$$\langle u, v \rangle_E = \sum_{i=1}^{n} (x_{i+1/2} - x_{i-1/2}) u_i v_i$$

2. Vérifier que

$$(D^\star v)_i = -\frac{v_{i+1/2} - v_{i-1/2}}{x_{i+1/2} - x_{i-1/2}} \qquad \text{pour tout } 1 \le i \le n.$$

3. Vérifier que $\mathcal{D}^\star = -\mathcal{D}$

2 La programmation orientée objet

(15 mins) Exercice 2.1

J'ai effacé une partie d'un programme que vous devez re-écrire, vous devez cependant comprendre tout le programme pour arriver à rétablir la partie manquante à la place du commentaire `##TODO` dans la fonction `adjoint()`.

```python
class my_strange_linear_op() :
    def __init__(self,input_len) :
        assert isinstance(input_len,int)
        assert input_len > 2
        self.input_len=input_len
        self.output_len=2*input_len
        np.random.seed(42)
```

```
        self.A=np.random.randn(self.output_len,self.input_len)
    def direct(self,x) :
        assert x.shape==(self.input_len,)
        u=self.A@x
        u[0]=4*u[0]
        u[1]+=u[0]
        return u
    def adjoint(self,u) :
        assert u.shape==(self.output_len,)
        ##TODO
    def test(self) :
        np.random.seed(42)
        x=np.random.randn(self.input_len)
        u=np.random.randn(self.output_len)
        print("0 ?=? ",np.sum(self.direct(x)*u)-np.sum(x*self.adjoint(u)))
import numpy as np
e=my_strange_linear_op( 4 )
e.test() # 0 ?=?  -1.7763568394002505e-15
```

3 Approximation de fonctions

(10 mins) Exercice 3.1

Soit une suite $(I_\ell, O_\ell)_{\ell=1,\dots,L}$ un jeu de donnée avec I_ℓ et O_ℓ dans $\mathbb{R}$ pour tout ℓ. On suppose que tous les I_ℓ sont deux à deux différents. On cherche une fonction $f_\Theta : \mathbb{R} \to \mathbb{R}$ avec $f(I) \simeq O$ tel que $f_\Theta(x) = \Theta_2 x^2 + \Theta_1 x + \Theta_0$. Ici notre ensemble de fonction est paramétré par $\Theta \in \mathbb{R}^3$. On utilisera la fonction perte $d(\hat{o}, o) = \frac{1}{2}|o - \hat{o}|^2$.

1. Montrez que le problème d'approximation est un problème de moindre carré linéaire. Ecrire les équations normales.
2. Dans le cas $L = 3$, comment s'appelle le problème ? dans le cas $L < 3$ y-a-t'il existence et unicité des solutions ?

4 Optimisation d'un réseau de neurones

(20 mins) Exercice 4.1

On s'intéresse à la fonction suivante de $\theta = (\theta_0, \theta_1, \theta_2) \in \mathbb{R}^3$

$$x_4(\theta) = \sqrt{\theta_2^2 + (\theta_1 \cos(\theta_0 + 2))^2}$$

- Calculez directement le gradient de $\theta \mapsto x_3(\theta)$.
- On définit $\mathcal{F}_0(x_0, \theta_0) = \cos(\theta_0 + x_0)$, $\mathcal{F}_2(x_2, \theta_2) = \theta_2^2 + x_2^2$. Définir $\mathcal{F}_1$ et $\mathcal{F}_3$ pour que x_4 soit défini par la récurence suivante

  $$x_{s+1} = \mathcal{F}(x_s, \theta_s) \quad x_0 = 2$$

  Vous donnerez les formules de $X = (x_i)_{i=0,;4}$
- Utiliser la formule de rétropropagation de gradient pour trouver le gradient de x_4 par rapport à θ (on exprimera tout en fonction de $X = (x_i)_{i=0,\ldots,4}$). Comparez avec la méthode de la première question.

(15 mins) Exercice 4.2: Gradient implicite

On s'intéresse à la fonction $y = \mathcal{F}(x, \theta)$ où $x \in \mathbb{R}^n$, $\theta \in \mathcal{M}_{nn}(\mathbb{R})$ tel que θ^{-1} existe et y est défini de manière implicite par

$$\theta y = x.$$

1. Pour tout $\dot{\theta}$ et $\dot{x}$, donnez la formule de $\dot{y} = (\partial_x \mathcal{F})\dot{x} + (\partial_\theta \mathcal{F})\dot{\theta}$
2. Pour tout $\hat{y}$, donnez la formule de $\hat{\theta} = (\partial_\theta \mathcal{F})^\star \hat{y}$ et $\hat{x} = (\partial_x \mathcal{F})^\star \hat{y}$

5 Exemple d'annale

(40 mins) Exercice 5.1

On se donne la couche $y = \mathcal{F}(x, \theta)$ définie pour tout $x \in \mathcal{M}_{NL}(\mathbb{R})$, $\theta = (A, B)$ avec $A \in \mathcal{M}_{MN}(\mathbb{R})$ et $B \in \mathbb{R}^M$ par

$$y_{ml} = \sum_{n=1}^{N} A_{mn}^2 \cos(x_{n\ell}^2) + B_m \text{ pour tout } 1 \leq \ell \leq L \text{ et } 1 \leq m \leq M.$$

Ainsi y est une matrice de $\mathcal{M}_{ML}(\mathbb{R})$.

1. Si $\dot{x}$ et $\dot{\theta} = (\dot{A}, \dot{B})$ sont donnés, donnez la formule de $\dot{y} = (\partial_x \mathcal{F})\dot{x} + (\partial_\theta \mathcal{F})\dot{\theta}$.
2. Pour tout $\hat{y}$, donnez la formule de $\hat{\theta} = (\partial_\theta \mathcal{F})^\star \hat{y}$ et $\hat{x} = (\partial_x \mathcal{F})^\star \hat{y}$. On notera $\hat{\theta} = (\hat{A}, \hat{B})$.
3. On note ci-dessous une classe `interro` qui implémente cette couche. Donnez les lignes de code qu'il faut mettre à la place des balises `#TODO1` à `#TODO6` pour implémenter correctement cette couche. On rappelle la structure de la classe `Parameter` et de la classe `Dense` en préambule.

```
import numpy as np

class Parameter() :
    def __init__(self,shape) :
```

```
        self.shape=shape
        np.random.seed(42)
        self.grad=np.zeros(self.shape)
        self.size=self.grad.size
        self.data=np.random.randn(self.size).reshape(self.shape)

class Dense() :
    def __init__(self,nb_entree,nb_sortie) :
        A=Parameter((nb_sortie,nb_entree))
        b=Parameter((nb_sortie,1))
        self.list_params=[A,b] # Liste des paramètres
        self.save=None # Objet pour sauver des infos dans le forward
    def forward(self,x) :
        self.save=np.copy(x)
        (A,b)=[p.data for p in self.list_params]
        y=A@x+b
        return y
    def backward(self,hat_y) :
        x=self.save
        (A,b)=[p.data for p in self.list_params]
        hat_A=hat_y@x.T
        hat_b=np.sum(hat_y,axis=1)
        hat_x=A.T@hat_y
        self.list_params[0].grad=hat_A
        self.list_params[1].grad=hat_b
        for p in self.list_params :
            p.grad=p.grad.reshape(p.shape)
        return hat_x

class Interro()
  def __init__(self,nb_entree,nb_sortie) :
        self.list_params=[Parameter((nb_sortie,nb_entree)),Parameter((nb_sortie,1))]
        self.save=None
    def forward(self,x) :
        self.save=np.copy(x)
        (A,b)=[p.data for p in self.list_params]
        y=     None #TODO1 : change this
        return y
    def backward(self,hat_y) :
        x=     None #TODO2 : change this
        (A,b)= None #TODO3 : change this
        hat_A= None #TODO4 : change this
        hat_b= None #TODO5 : change this
        hat_x= None #TODO6 : change this
        self.list_params[0].grad=hat_A
        self.list_params[1].grad=hat_b
        for p in self.list_params :
            p.grad=p.grad.reshape(p.shape)
        return hat_x
```

INSA TOULOUSE
135 avenue de Rangueil
31400 Toulouse

Tél : + 33 (0)5 61 55 95 13
www.insa-toulouse.fr

INSTITUT NATIONAL
DES SCIENCES
APPLIQUÉES
TOULOUSE