Keras

Introduction à l'apprentissage automatique
Master Sciences Cognitives
Aix Marseille Université

Alexis Nasr

Introduction

- `keras` est une librairie `python` qui permet de manipuler des réseaux de neurones
- L'utilisation de `keras` suppose en général les étapes suivantes :
  1. Préparation des données
  2. Construction d'un modèle
  3. Estimation des paramètres du modèle
  4. Evaluation du modèle

Préparation des données

- Les données doivent se trouver sous la forme de tableaux `numpy`
- Ces tableaux sont généralement bi-dimensionnels : $n \times d$ où n correspond au nombre d'exemples et d à la dimension de chaque exemple.
- Exemple :
  - `x_train = np.array([[4.79, 0.02], [2.78, 1.9], [4.98,0.1], [2.51, 2.03], [5.24, 0.12], [2.63, 1.6]])`
  - `y_train = np.array([[0,1], [1,0], [0,1], [1,0], [0,1], [1,0]])`

Préparation des données

- En général les données ne sont pas intégrées dans le programme python, mais elles sont lues depuis des fichiers de données.
  - Entrées :

```
4.79    0.02
2.78    1.9
4.98    0.1
2.51    2.03
5.24    0.12
2.63    1.60
```

  - Sorties :

```
fr
en
fr
en
fr
en
```

Fonction de lecture des données

```
def lectureEntrees(nomFichier):
    try:
        fic = open(nomFichier, 'r')
    except IOError:
        print("le fichier", nomFichier, "n'existe pas")
        return None

    lx = []
    for ligne in fic:
        ligne = ligne.strip('\n\r')
        liste = ligne.split()
        lx.append([float(x) for x in liste])
    fic.close()
    return(np.array(lx, dtype="float"))
```

Fonction de lecture des données

```
def lectureSorties(nomFichier):
    try:
        fic = open(nomFichier, 'r')
    except IOError:
        print("le fichier", nomFichier, "n'existe pas")
        return None

    codeClasse = { }
    ly = []

    for ligne in fic:
        ligne = ligne.strip('\n\r')
        if not ligne in codeClasse :
          codeClasse[ligne] = len(codeClasse)
        ly.append(codeClasse[ligne])

    lyOneHot = []
    dimension = len(codeClasse)
    for code in ly :
        lyOneHot.append(encodeOneHot(code, dimension))
    return(np.array(lyOneHot, dtype="int"))

def encodeOneHot(elt, size):
    vect = [0] * size
    vect[elt] = 1
    return vect
```

Construction du modèle

```
def construitModele(dimEntree, dimSortie):
    modele = Sequential()
    modele.add(Dense(units=16, activation='relu', input_dim=2))
    modele.add(Dense(units=2, activation='softmax'))
    modele.compile(loss='categorical_crossentropy',
                   optimizer='adam',
                   metrics=['accuracy'])
    return(modele)
```

Affichage de la structure du modèle

```
modele = construitModele(2, 2)
print(modele.summary())
```

```
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 16)                48
_________________________________________________________________
dense_1 (Dense)              (None, 2)                 34
=================================================================
Total params: 82
Trainable params: 82
Non-trainable params: 0
_________________________________________________________________
```

Apprentissage des paramètres

```
modele.fit(x_train, y_train, epochs=10, batch_size=1, validation_split=0.2)
Epoch 1/10
16/16  - loss: 1.7136 - accuracy: 0.5000 - val_loss: 1.5570 - val_accuracy: 0.5000
Epoch 2/10
16/16  - loss: 1.4296 - accuracy: 0.5000 - val_loss: 1.3181 - val_accuracy: 0.5000
Epoch 3/10
16/16  - loss: 1.1962 - accuracy: 0.5000 - val_loss: 1.0933 - val_accuracy: 0.5000
Epoch 4/10
16/16  - loss: 0.9845 - accuracy: 0.5000 - val_loss: 0.8795 - val_accuracy: 0.5000
Epoch 5/10
16/16  - loss: 0.7917 - accuracy: 0.5000 - val_loss: 0.6999 - val_accuracy: 0.5000
Epoch 6/10
16/16  - loss: 0.6204 - accuracy: 0.5000 - val_loss: 0.5799 - val_accuracy: 0.5000
Epoch 7/10
16/16  - loss: 0.5275 - accuracy: 0.5000 - val_loss: 0.4534 - val_accuracy: 0.5000
Epoch 8/10
16/16  - loss: 0.4051 - accuracy: 0.8125 - val_loss: 0.3839 - val_accuracy: 1.0000
Epoch 9/10
16/16  - loss: 0.3423 - accuracy: 1.0000 - val_loss: 0.3261 - val_accuracy: 1.0000
Epoch 10/10
16/16  - loss: 0.2897 - accuracy: 1.0000 - val_loss: 0.2844 - val_accuracy: 1.0000
```