TD : `keras` & perceptron multi-couches

Romain Tavenard

Dans cette séance, nous nous focaliserons sur la création et l'étude de modèles de type perceptron multi-couches à l'aide de la librairie `keras`.

Pour cela, vous utiliserez la classe de modèles `Sequential()` de `keras`. Voici ci-dessous un exemple de définition d'un tel modèle :

```
from keras.models import Sequential
from keras.layers import Dense

#1. définir les couches et les ajouter l'une après l'autre au modèle
premiere_couche = Dense(units=12, activation="relu", input_dim=24)
couche_cachee1 = Dense(units=12, activation="sigmoid")
[..]
couche_sortie = Dense(units=3, activation="linear")

model = Sequential()
model.add(premiere_couche)
model.add(couche_cachee1)
[...]
model.add(couche_sortie)

#2. Spécifier l'algo d'optimisation et la fonction de risque à optimiser
# Fonctions de risque classiques :
# * "mse" en régression,
# * "categorical_crossentropy" en classification multi-classes
# * "binary_crossentropy" en classification binaire
# On peut y ajouter des métriques supplémentaires (ici taux de bonne
# classifications)

model.compile(optimizer="sgd", loss="mse", metrics=["accuracy"])

#3. Lancer l'apprentissage
model.fit(X_train, y_train, verbose=2, epochs=10, batch_size=200)
```

1 Préparation des données

Pour ce TD, un module `dataset_utils` vous est fourni sur CURSUS. Il permet de récupérer des jeux de données sur lesquels s'entrainer en TD. Dans ce TD, nous travaillerons sur le jeu MNIST, dans lequel les exemples fournis sont des chiffres écrits à la main et la classe à prédire est le chiffre effectivement représenté.

Dans le module `dataset_utils`, une fonction `prepare_mnist` est fournie à laquelle on doit juste passer un booléen indiquant si l'on travaille depuis une machine du réseau MASS (auquel cas les jeux de données sont déjà téléchargés) ou une machine perso.

1. Observez le code des fonctions `prepare_mnist` et `prepare_boston`. Que font ces fonctions ? Quelles sont les dimensions des matrices / vecteurs à la sortie ? S'agit-il de problèmes de classification ou de régression ?

2 Premiers réseaux de neurone

2. Chargez le jeu de données MNIST et apprenez un premier modèle sans couche cachée avec une fonction d'activation raisonnable pour les neurones de la couche de sortie. Pour cette question comme pour les suivantes, limitez vous à un nombre d'itérations de l'ordre de 10 : ce n'est absolument pas réaliste, mais cela vous évitera de perdre trop de temps à scruter l'apprentissage de vos modèles.
3. Comparez les performances de ce premier modèle è celle de modèles avec respectivement 1, 2 et 3 couches cachées de 128 neurones chacune. Vous utiliserez la fonction ReLU (`"relu"`) comme fonction d'activation pour les neurones des couches cachées.
4. On peut obtenir le nombre de paramètres d'un modèle à l'aide de la méthode `count_params()`. Comptez ainsi le nombre de paramètres du modèle à 3 couches cachées et définissez un modèle à une seule couche cachée ayant un nombre comparable de paramètres. Parmi ces deux modèles, lequel semble le plus performant ?

3 Utilisation d'un jeu de validation

Bien entendu, les observations faites plus haut ne sont pas suffisantes, notamment parce qu'elles ne permettent pas de se rendre compte de l'ampleur du phénomène de sur-apprentissage.

Pour y remédier, `keras` permet de fixer, lors de l'appel à la méthode `fit()`, une fraction du jeu d'apprentissage à utiliser pour la validation. Jetez un oeil ici pour comprendre comment les exemples de validation sont choisis.

5. Répétez les comparaisons de modèles ci-dessus en vous focalisant sur le taux de bonnes classifications obtenu sur le jeu de validation (vous prendrez 30% du jeu d'apprentissage pour votre validation).

4 Régularisation et *Drop-Out*

6. Appliquez une régularisation de type L_1 à chacune des couches de votre réseau. L'aide disponible ici devrait vous aider.
7. Au lieu de la régularisation L_1, choisissez de mettre en place une stratégie de *Drop-Out* pour aider à la régularisation de votre réseau. Vous éteindrez à chaque étape 10% des poids de votre réseau.