Keras – Tensorflow
Regression et Optimisation

Housing Californie

Les données sur le logement en Californie sont disponibles en utilisant la méthode fetch_california_housing () de sklearn.datasets.

La méthode retourne un objet avec un attribut DESCR décrivant l'ensemble de données, un attribut « data » avec les entités en entrée et un attribut « target » avec les labels.

L'objectif est de prédire le prix des maisons dans un district (un bloc de recensement) en fonction de statistiques sur ce district en utilisant un réseau de neurones.

Etapes :

- Train-test
- StandardScaler
- Construction, entraînement et évaluation d'un réseau de neurones.
- Evaluation et affichage de la courbe d'apprentissage.

Optimisation des hyperparamètres

1. Entraîner le modèle plusieurs fois, en faisant varier le « learning rate » (par exemple, 1e-4, 3e-4, 1e-3, 3e-3, 3e-2) et comparez les courbes d'apprentissage.

2. Il est possible, dans un objectif d'optimisation, de créer une fonction build_model () qui prend trois arguments, n_hidden, n_neurons, learning_rate, et qui construit, compile et retourne un modèle avec n_hidden couches cachées, n_neurons de neurones et le taux d'apprentissage donné. Les arguments seront initialisés à des valeurs par défaut.

   La fonction build_model servira alors d'argument à un objet keras.wrappers.scikit_learn.KerasRegressor. (https://keras.io/scikit-learn-api/)
   Cette méthode permet de créer un prédicteur compatible Scikit-Learn.

   Créer un KerasRegressor et entraîner le modèle en précisant les n_epochs, les callbacks et les données de validation en arguments de la méthode fit().

3. sklearn.model_selection.RandomizedSearchCV permet également d'effectuer une optimisation des hyperparamètres du KerasRegressor. Il nécessite la définition d'un intervalle ou d'un ensemble pour chaque hyperparamètre à étudier.
   La doc : https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html
   Voir les étapes ci-dessous.

Etapes :

- créer un dictionnaire params où chaque clé est le nom d'un hyperparamètre à ajuster ("n_hidden", …) et chaque valeur est la liste des valeurs à explorer (par exemple, [0, 1, 2, 3]), ou une distribution ce scipy.stats.
  Ainsi, le learning rate poura être donné par une distribution reciproque (par exemple, reciprocal (3e-3, 3e-2)).
- Créez un RandomizedSearchCV, en passant le KerasRegressor et les params au constructeur, ainsi que le nombre d'itérations (n_iter) et le nombre de cross validation(cv). Exemple n_iter = 10 et cv = 3.
- Entraîner ensuite le modèle en appelant la méthode fit() en précisant `n_epochs`, `validation_data` et `callbacks`
- Les meilleurs paramètres seront disponibles dans best_score_ et le meilleur modèle sera dans best_estimator_.

Conseil : Explorer d'autres méthodes d'optimisation telle que :

- Hyperopt
- Hyperas
- Sklearn-Deap
- Scikit-Optimize
- Spearmint
- PyMC3
- GPFlow
- Yelp/MOE
- Mais aussi Google Cloud ML Engine, Arimo ou Oscar