Manipulation Deep Learning

Alison PATOU
17/12/2021

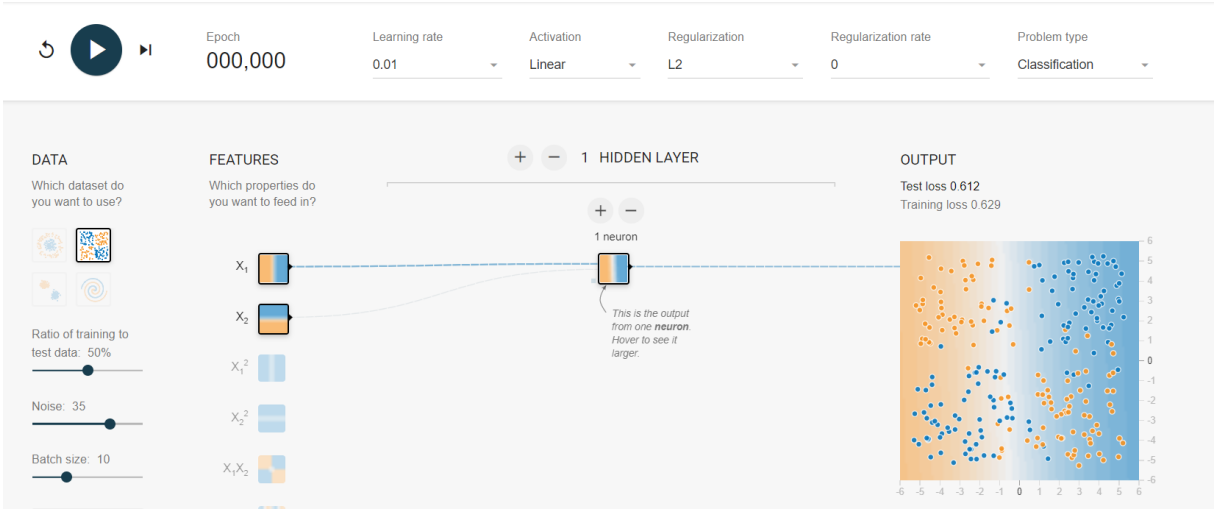
Partie 1

L'idée de ce premier TP est de tester la manipulation du Playground de Tensorflow disponible à cette adresse :

https://playground.tensorflow.org/

Nous allons donc pouvoir entrainer notre premier petit réseau de neurones. Les réseaux de neurones permettent d'apprendre des modèles non linéaires sans avoir à recourir à des croisements de caractéristiques explicites.

Configurez votre petit réseau de neurones de la même manière que sur la copie d'écran suivante :



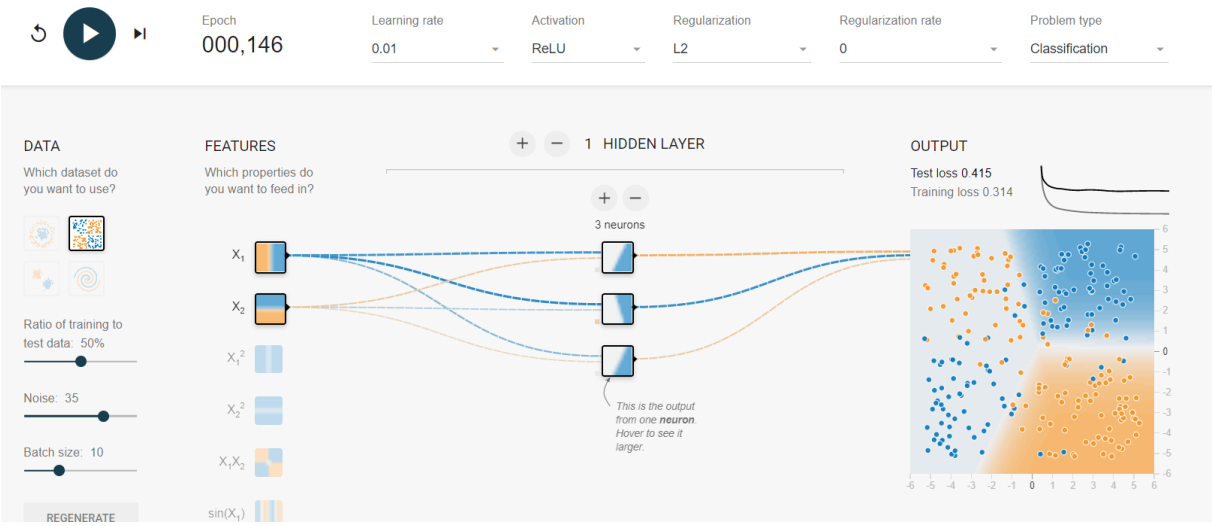
1. Le modèle contient des combinaisons données de deux caractéristiques d'entrées dans un seul neurone. Ce modèle peut-il apprendre des non-linéarités ? Exécutez-le pour vérifier votre intuition.

2. Changez maintenant le nombre de neurones dans la couche cachée en modifiant 1 par 2, et remplacez l'activation linéaire par une activation non linéaire, par exemple ReLU. Pouvez-vous créer un modèle capable d'apprendre des non-linéarités ?

3. Poursuivez l'expérience en ajoutant ou en supprimant des couches cachées et des neurones dans les couches. N'hésitez pas non plus à modifier les taux d'apprentissage, la régularisation et les autres paramètres d'apprentissage. Quel est le plus petit nombre de nœuds et de couches que vous pouvez utiliser et qui donne une perte d'évaluation de 0,177 ou moins ?

Partie 2

Nous allons maintenant complexifier un peu notre démarche. Configurez votre petit réseau de neurones de la même manière que sur la copie d'écran suivante :

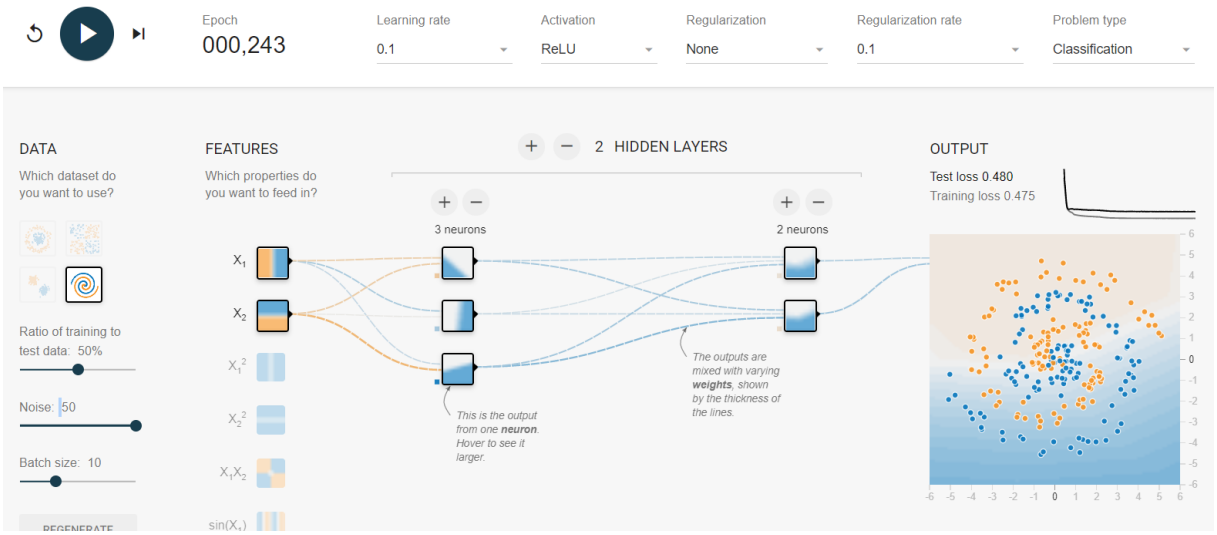


1. Exécutez le modèle donné quatre ou cinq fois. Avant chaque essai, cliquez sur le bouton Réinitialiser le réseau pour obtenir une nouvelle initialisation aléatoire. (Le bouton Réinitialiser le réseau est la flèche circulaire de réinitialisation qui se trouve juste à gauche du bouton "Lecture".) Laissez l'essai exécuter au moins 500 pas pour assurer la convergence. Vers quelle forme le résultat de chaque modèle converge-t-il ? Que peut-on en déduire sur le rôle de l'initialisation dans une optimisation non convexe ?

2. Essayer d'augmenter légèrement la complexité du modèle en ajoutant une couche et quelques nœuds. Répétez les essais de la tâche 1. Les résultats sont-ils plus stables ?

Partie 3

Nous allons maintenant construire un modèle de réseaux de neurone en spiral.



1. Entraînez le meilleur modèle que vous pouvez en utilisant uniquement X1 et X2. N'hésitez pas à supprimer ou à ajouter des couches et des neurones, et à changer les paramètres d'apprentissage tels que le taux d'apprentissage, le taux de régularisation et la taille du lot. Quelle est la meilleure perte d'évaluation que vous pouvez obtenir ? La surface de sortie du modèle est-elle lisse ?

2. Même avec les réseaux de neurones, un certain niveau d'extraction de caractéristiques est nécessaire pour obtenir les meilleures performances possibles. Essayez d'ajouter des caractéristiques de produits croisées et d'autres transformations telles que sin(X1) et sin(X2). Le modèle est-il plus performant ? La surface de sortie du modèle est-elle plus lisse ?

Partie 4 (Optionnel en Python)

Dans cette partie, nous travaillerons sur le jeu de données CIFAR-10, un classique dans le milieu du Deep Learning.

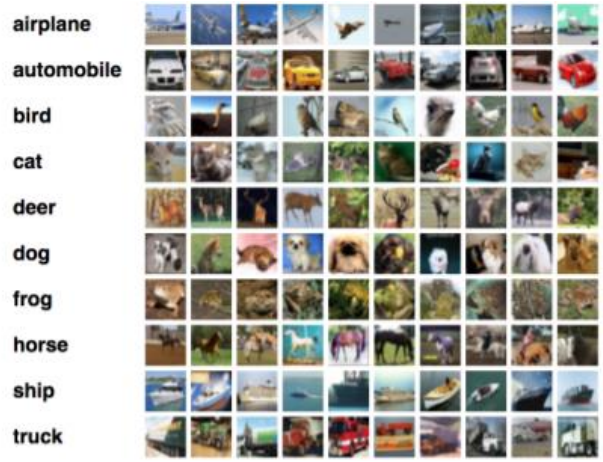


FIGURE 3 – Base de données CIFAR-10.

Cette base d'images RGB de 32×32 pixels comporte 10 classes, 50k images en train et 10k images en test.

Le réseau que nous allons implémenter a un style proche de l'architecture AlexNet de Krizhevsky et al. (2012) adaptée à la base CIFAR-10 dont les images sont plus petites.
Il sera composé des couches suivantes :

- conv1 : 32 convolutions 5×5, suivie de ReLU
- pool1 : max-pooling 2×2
- conv2 : 64 convolutions 5×5, suivie de ReLU
- pool2 : max-pooling 2×2
- conv3 : 64 convolutions 5×5, suivie de ReLU
- pool3 : max-pooling 2×2
- fc4 : fully-connected, 1000 neurones en sortie, suivie de ReLU
- fc5 : fully-connected, 10 neurones en sortie, suivie de softmax

Implémentez ce modèle en Python et évaluez-le.
Note : Le code suivant vous permettra de charger les données :

```
# example of loading the cifar10 dataset
from matplotlib import pyplot
from keras.datasets import cifar10
# load dataset
(trainX, trainy), (testX, testy) = cifar10.load_data()
```