Intelligence Artificielle : feuille 3

Deep learning
Régression linéaire, classification

Présentation

Le but de ce premier TD est de vous familiariser avec les réseaux de neurones et leur entraînement. Nous allons passer en revue les éléments de base suivant :

– données d'apprentissage,

– fonction de perte (loss function),

– descente de gradient.

Pour cela, nous allons utiliser un réseau de neurones (composé d'un seul neurone en fait) pour deux tâches :

– une régression linéaire simple

– une classification binaire.

Pour la réalisation de ce TD, vous avez le choix entre télécharger le fichier `dl_td1.py` ou télécharger le notebook `dl_td1.ipynb` et compléter l'un ou l'autre. Si vous optez pour le notebook, gardez en tête que les cellules sont plus ou moins indépendante et que, selon les cas, il se peut que vous ayez à ré-exécuter une cellule pour qu'une autre, situées plus loin puisse donner un résultat.

Exercice 1. Un neurone

Comme vu en cours, un neurone formel est représenté comme suit :

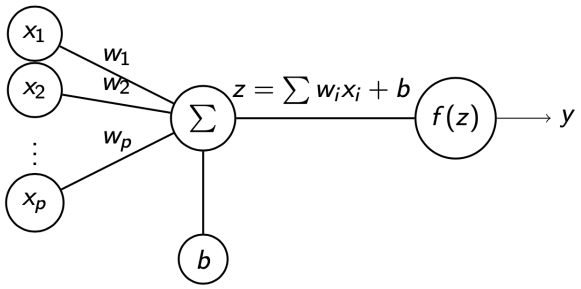

FIGURE 1 – Un neurone formel (Mc Culloch & Pitts (1943))

1. Nommez chacun des éléments du neurone.
2. Ecrivez une fonction `neuron(x, w, b, f)` permettant d'implémenter ce neurone
3. Définissez les deux fonctions `id` et `sigmoid` dont l'expression est :

$$\texttt{id}(z) = z \text{ et } \texttt{sigmoid}(z) = \frac{1}{1+e^{-z}}.$$

4. Testez votre code.

Exercice 2. Régression linéaire

Dans cet exercice, nous allons entraîner un neurone pour réaliser une régression linéaire simple.

1. Dessiner le neurone qui sera utilisé. Identifiez bien les paramètres à trouver.
2. Ecrivez le code de la fonction `loss(y, y_hat)` définie comme la moyenne des carrés des écarts divisée par 2 :
$$\text{loss}(y, \hat{y}) = \frac{1}{2n} \sum_{i=1}^{n} (\hat{y_i} - y_i))^2 .$$
3. Calculez le gradient de la fonction $\mathcal{L} = \text{loss}$ par rapport aux deux variables w et b.
4. Ecrivez la fonction `gradient(x, y_hat, y)` retournant le gradient de la fonction $\mathcal{L}$.
5. Ecrivez une fonction `train(x, y, eta, epochs)` qui réalise une descente du gradient de la fonction $\mathcal{L}$ afin de trouver les paramètres w et b minimisant la valeur de $\mathcal{L}$.
6. Entraînez votre neurone. Prenez 0.001 comme valeur pour le pas d'apprentissage, et 10000 pour le nombre d'epochs.
7. Donnez l'équation de la droite de régression obtenue. Ecrivez l'instruction permettant de calculer les valeurs prédites.
8. Dessinez la droite de régression sur la même figure que le nuage de point.
9. Comparez le résultat obtenu avec la droite donnée par `linregress` du sous module `stats` du module`scipy`.

Exercice 2. Régression logistique

Dans cet exercice, nous allons transformer le neurone afin de l'utiliser pour réaliser une classification. Nous allons nous limiter à une classification binaire : les éléments appartiennent soit à une classe 0 soit à une classe 1.

1. Comment devons-nous appeler la "fonction" `neuron` pour que le fonction d'activation soit la sigmoid ?
2. En utilisant la fonction `make_classification`, (voir le code), générez les données, et découpez le corpus en deux parties train et test.
3. Définissez la nouvelle fonction de perte. Nous allons utiliser la fonction *cross_entropy* vue en cours :
$$\mathcal{L} = -\frac{1}{n} \sum_{i=1} y_i \times \log(\hat{y}_i)$$
4. Ecrivez les expressions mathématiques de ce que calcule votre neurone.
5. Calculez le gradient de $\mathcal{L}$ et écrivez la fonction correspondante
6. Adaptez une fonction `train(x, y, eta, epochs)` pour réaliser une descente du gradient de la fonction $\mathcal{L}$ afin de trouver les paramètres w et b minimisant la valeur de $\mathcal{L}$.
7. Entraînez votre neurone. Exécutez d'abord le code fourni pour mettre au bon format le vecteur en entrée.
8. Evaluez votre modèle.