Semaine IMT Grand-Est
Introduction à l'apprentissage automatique
Séance 4
Exercice 1: *Playground Tensorflow*

Dans ce premier exercice, nous discutons et illustrons différentes propriétés fondamentales des perceptrons multi-couches à l'aide du « bac à sable » de TensorFlow disponible ici :

`playground.tensorflow.org`

Voici une brève description de l'interface.

- Le bandeau du haut permet les réglages suivants, de gauche à droite :
  - relancer l'apprentissage, lancer l'apprentissage, exécuter un pas (une *epoch*) d'apprentissage (`Epoch` indique l'avancée de l'apprentissage) ;
  - changer le *learning rate* (il s'agit du pas dans la descente de gradient) ;
  - la fonction d'activation (vous avez le choix) ;
  - le type et le taux de régularisation (cf slides) ;
  - le type de problème avec lequel on joue (restez sur `Classification`).
- le panneau du bas donne les indications suivantes, de gauche à droite :
  - `Data` : choix d'une base de donnée dans le plan, proportion entre apprentissage et test, bruit sur les données, taille du lot (apprentissage par lot, cf slides), bouton pour régénérer les données.
  - `Features`, il s'agit du choix des neurones d'entrée : par défaut les deux coordonnées de points du plan, mais on peut ajouter le carré de chaque coordonnée, leur produit, ou leur sinus. Chaque carré représente les valeurs des entrées pour les points du domaine couvrant la base de données.
  - Architecture du réseau : on peut changer le nombre de couches cachées et le nombre de neurones dans chaque couche. Les poids des liaisons sont représentés par différentes épaisseur et code couleur, un survol avec la souris permet d'afficher la valeur courante, et même de la changer. Le carré en chaque neurone représente la sortie du neurone pour chaque valeur de la couche précédente : dans la première couche cachée, il s'agit des sorties pour les valeurs d'entrée lorsque celles-ci parcourent le domaine de la base de données.
  - `Output` : représentation graphique du *loss* calculé sur la base de test, du *loss* calculé sur la base d'apprentissage (il est minimisé par descente de gradient) ; représentation de la sortie du réseau en chaque point du domaine, superposé aux données d'apprentissage ; échelle de couleurs représentant les différentes valeurs ; et possibilité de montrer les données test et de discrétiser la sortie (seuil par rapport à 0, c'est la fonction de décision pour une classification bi-classe).

Vous reviendrez au paramètres par défaut (rechargez la page) après chaque question.

Remarque générale : on peut avoir envie d'utiliser une valeur plus grande du *learning rate* pour accélérer la convergence de l'algorithme de descente de gradient à pas fixe. Néanmoins, lorsque le graphique du *training loss* se met à osciller en restant globalement au même niveau, il faut diminuer le *learning rate*.

Faites les expériences suivantes (et n'hésitez pas à expérimenter librement) :

1. Commencez par une architecture à 0 couche cachée, et cochez *discretize output* : il s'agit du perceptron de Rosenblatt (cependant, l'apprentissage ne se fait pas avec l'algorithme du perceptron vu en cours mais bien par minimisation du *training loss*). Constatez que le perceptron ne peut apprendre que des séparations linéaires, ce qui est utile pour une seule des bases de données proposées.

2. Utilisez à présent une couche cachée composée d'un neurone. Lancez l'apprentissage, et visualisez la forme de la séparation.

3. Utilisez à présent une couche cachée composée de deux neurones. Lancez l'apprentissage, et visualisez la forme de la séparation sur les différentes bases de donnés.
   Que se passe-t-il lorsqu'on augmente le nombre de neurones ?

4. Toujours avec une couche cachée, utilisez l'activation ReLU. Comment évoluent les surfaces de séparation ?
   Que se passe-t-il si on choisit une activation *linear* ?

5. Ajoutez une deuxième couche cachée, commencez par deux neurones dans chaque couche cachée, et lancez l'apprentissage sur chaque base.

6. Constatez que si on enrichit les *features* en entrée, on peut se contenter d'un réseau bien plus simple.

7. On considère la base de données formée des deux spirales entremêlées. Plutôt qu'ajouter des *features* non linéaire et utiliser un réseau simple, on revient aux *features* X^1 et X^2, et on considère un réseau de 6 couches cachées à 8 neurones chacune, fonction d'activation ReLU (plus rapide à calculer que $\tanh$ ou sigmoïde), *learning rate* de 0.03 (au départ), régularisation L^2, et *regularization rate* de 0.003. Laissez l'apprentissage se dérouler pendant quelques centaines d'*epochs* (en diminuant le *learning rate* en cours d'apprentissage, comme expliqué dans la remarque préliminaire).

8. On revient aux paramètres par défaut. Augmentez la valeur de *noise* à 50. Les deux classes sont alors assez entremêlées. Constatez que l'entraînement du réseau mène au surapprentissage (vous pouvez commencer avec un *learning rate* à 0.3). Comment l'éviter ?