Université de Lille
M2 ISN/FC

Enseignant : Pascal Germain
26 octobre 2018

Projet du cours *Introduction aux réseaux de neurones*

À réaliser en équipe de **2 ou 3 étudiants**
Date de remise : **31 janvier 2019**

Remise du projet

Envoyez un fichier compressé à `pascal.germain@inria.fr` contenant :

- Un fichier « `Noms.txt` » contenant les noms des coéquipiers.
- Un rapport de **5 à 10 pages en format PDF** (LaTeX recommandé)
- Le **code source** permettant de reproduire vos expérimentations, sous la forme de code source python (extension «`.py`») et/ou de carnets jupyter-notebooks (extension «`.ipynb`»).

Barème d'évaluation

Ce projet vaut pour 60% de la note finale du cours de réseau de neurones. Le projet sera évalué comme suit.

- Partie 1 : 7 points pour le rapport et 5 points pour le code.
- Partie 2 : 5 points pour le rapport et 3 points pour le code.

Notez bien

Ce projet a pour objectif de vous familiariser avec la mise en oeuvre des réseaux de neurones. Si vous désirez réaliser votre projet sur autre application que celle suggérée ici, je vous invite à venir en discuter avec moi.

Le jeu de données et le code fourni

Tout le matériel relié au projet se trouve dans le répertoire «`travail_pratique`» du github du cours.

L'ensemble CIFAR. Nous vous fournissons un sous-ensemble du jeu de données CIFAR10 [1]. Il s'agit d'un problème de classification multi-classes ; le jeu de données contient des images couleurs de taille 32×32 pixels représentant 10 catégories d'objets. Pour simplifier le problème et réduire le temps d'apprentissage requis, nous vous suggérons de conserver seulement les trois premières catégories : «avion», «automobile» et «oiseau».

Le répertoire «`cifar`» contient un fichier compressé par catégorie [2], chacun regroupant les images en format PNG.

Le code fourni. Nous vous fournissons une collection de méthodes et de classes dans le fichier «`projetutils.py`». En particulier, la méthode «`charger_cifar`» permet d'extraire les images compressées du jeu de données et de les transformer en vecteur de $3 \times 32 \times 32 = 3072$ nombres réels compris entre 0.0 et 1.0, qui sont la concaténation des valeurs des canaux rouge, vert et bleu.

L'usage des méthodes de «`projetutils.py`» est présenté dans le carnet jupyter-notebook «`Projet-demo.ipynb`». Ce carnet est aussi le point de départ de la partie 1 du projet.

1. Le jeu de donnée original provient de : `https://www.cs.toronto.edu/~kriz/cifar.html`
2. Bien que nous vous suggérons de concentrer vos efforts sur le problème de classification constitué des trois premières catégories, nous vous fournissons les images des 10 catégories. Les étudiants curieux pourront y jeter un coup d'oeil, et les ambitieux pourront intégrer plus de trois catégories à leur apprentissage.

Partie 1

En guise de première approche pour résoudre le problème de classification à trois classes décrit plus haut, nous vous fournissons le code d'une architecture à une seule couche cachée de 50 neurones pleinement connectés. En comptant les deux matrices de poids ce réseau ainsi que les biais, ce réseau est formé de 153 803 paramètres, c'est à dire :

— Couche cachée : [3 072 entrées] × [50 neurones] + [50 valeurs de biais] = 153 650 paramètres.
— Couche de sortie : [50 entrées] × [3 neurones] + [3 valeurs de biais] = 153 paramètres.

Vous constaterez dans le carnet jupyter-notebook fourni avec cet énoncé que ce réseau possède une précision d'environ 72% sur le jeu de données fourni.

Nous vous demandons de suggérer **deux autres architectures** de réseau de neurones comportant un nombre similaire de paramètres à optimiser (on vous permet une différence approximative de 20%, soit environ de 120 000 à 180 000 paramètres). Au moins un de ces réseaux doit comporter des filtres de convolution.

Votre rapport doit présenter chacune de ces deux architectures en détaillant le **nombre de paramètres total à optimiser**, ainsi que la **répartition de ces paramètres dans chaque couche**. Vous devez **comparer les résultats** obtenus par vos deux réseaux entre eux, ainsi qu'avec le réseau pleinement connecté fourni avec l'énoncé. **Commentez les différences observées** en termes de précision sur les ensembles d'apprentissage et de test, mais aussi les autres impacts éventuels sur le nombre d'époques requises pour l'apprentissage et sur les ajustements requis aux paramètres de la descente de gradient (taille du pas, momentum, taille de la «minibach») pour *bien faire apprendre* vos architectures. Si possible, **proposez des explications** aux comportements observés. Si certaines observations empiriques vont à l'encontre de votre compréhension ou de votre intuition, alors **dites pourquoi les résultats obtenus vous étonnent**.

L'important ici n'est pas nécessairement d'obtenir la meilleure précision possible, mais plutôt de bien **analyser les résultats**. N'hésitez pas à proposer des architectures de réseaux de neurones originales, tout en expliquant pourquoi vous considérez qu'elles sont atypiques.

Partie 2

Reprenez une des deux architectures proposées lors de la Partie 1. On vous demande maintenant d'**analyser empiriquement l'impact d'un paramètre de configuration** de votre choix sur le jeu de données fourni. Par exemple, vous pouvez choisir une idée parmi les suivantes :

1. Ajouter une ou plusieurs composantes de «Dropout» au réseau (ou conserver le «dropout» si votre réseau en utilise déjà) et étudier l'influence du paramètre p (la probabilité d'inclure ou de retirer les neurones lors de l'optimisation).
2. Étudier l'influence de la taille du filtre de convolution.
3. Étudier l'influence de la taille de la «mini-batch».
4. Étudier l'influence de la taille de l'ensemble de validation lors de la procédure de «early stopping».

Interprétez vos résultats. De plus, illustrez **à l'aide d'un graphique** le comportement du réseau en fonction du paramètre de configuration étudié. Vous pouvez analyser la précision du réseau, mais aussi l'impact du paramètre de configuration étudié sur d'autres aspects liés—par exemple—au processus d'optimisation par descente de gradient.