UPMC
SORBONNE UNIVERSITÉS

M I 0 2 4

Projet IAD

Vers une Machine d'apprentissage persistante

Plan de développement

Auteur :
Étienne Simon

Encadrant :
Ludovic Denoyer

5 juillet 2014

Table des matières

1 Aperçu du PIAD 2
1.1 Objectifs 2
1.2 Références 2
2 Organisation du PIAD 2
2.1 Participants 2
2.2 Activités 2
2.3 Calendrier 4

1 Aperçu du PIAD

1.1 Objectifs

Ce PIAD est intitulé « Vers une Machine d'apprentissage persistante. » L'objectif sera d'écrire une plateforme pour l'apprentissage et le raisonnement sur des données hétérogènes. Les modèles ainsi appris devront être conservés dans une base de données en vue de leur réutilisation.

1.2 Références

L'ensemble du livrable est disponible sur un dépôt Mercurial, il est possible de le cloner avec la commande :

```
hg clone http://hg.esimon.eu/mi024
```

L'ensemble de la documentation y est présente, y compris ce document.

2 Organisation du PIAD

2.1 Participants

Ce PIAD est réalisé par un seul étudiant : Étienne Simon (numéro étudiant 3060576.) Il est encadré par Ludovic Denoyer (professeur au LIP6.)

2.2 Activités

Phase 1 : Bases

Interfaçage avec NMLP Ajout d'opérateurs de serialisation aux classes de NMLP, utilisation possible de Boost.Serialization. Il faudra ajouter les opérateurs sans modifier le code de NMLP et prendre soin à discerner le code des objets CPU du code des objets GPU ainsi que les contraintes qui y sont liées (e.g. espace mémoire plus limité en GPU.) Il peut être intéressant d'avoir un format d'archive portable.

Persistance Gestion d'une base de donnée des objets de NMLP en utilisant les opérateurs de serialisation écrits lors de la tâche précédente. Il faudra également stocker des métadonnées, par exemple un nom et une description pour les encodeur et les tests ou encore les paramètres d'apprentissage utilisés pour l'entraînement d'un modèle. Il sera intéressant d'avoir des identificateurs universels pour les modèles et les jeux de données (par exemple en les hachant).

SQLite est déjà présent dans le dépôt de NMLP. Pour s'assurer une plus grande flexibilité, une couche d'abstraction telle que SOCI sera utilisée.

Supervision Le superviseur est une interface pour manipuler les objets de NMLP présents dans la base de données. C'est lui qui se charge de l'apprentissage des modèles sur les données présentes dans la base de données. Il devra permettre l'exécution de commandes telles que « faire Z coups de gradients sur le modèle X avec le jeu de données Y. » Pour cella, il s'appuiera sur les algorithmes implémentés dans NMLP.

Test Après l'écriture du superviseur, il est possible de tester l'ensemble du code écrit jusqu'alors. Il sera question d'apprendre un modèle simple sur deux ou trois types de données différents.

Phase 2 : Développements

Contrôleur Le contrôleur devra présenter une interface plus haut niveau que le superviseur. En acceptant des commandes par un protocole réseau et en lançant le superviseur en tâche de fond par exemple. Le contrôleur pourra également fournir une interface pour l'ajout de données. Cela se réduira à l'interprétation de requêtes récupérées par Boost.Asio. Dans un premier temps, le serveur devra accepter une seule connexion. Une file de requête pourra être crée pour la gestion de connexions simultanées.

Parallélisation Enfin, une optimisation intéressante consisterait à lancer plusieurs superviseurs concurrentiellement, voire d'utiliser Boost.MPI pour paralléliser les opérations sur plusieurs machines.

Documentations et tests

Documentation développeur L'outil *de facto* standard d'écriture de documentation développeur pour C++ est Doxygen. Ce n'est pas une dépendance de NMLP, mais il est possible de générer la documentation séparément, cela ne devrait donc pas poser de problème.

Documentation utilisateur L'installation devrait être assez simple pour ne nécessiter rien de plus qu'un fichier INSTALL. Les binaires et leurs fonctionnalités devront cependant être décris entièrement.

Tests unitaires Les dépendances de NMLP offrent déjà un *framework* de test avec CTest et Boost.Test. Des tests unitaires seront écrit avec ces outils tout au long du développement.

2.3 Calendrier

Date	Semaine	Description
24/02	*0*	*Remise du cahier des charges et du plan de développement.*
25/02-31/03	**1..5**	**Phase 1 : Bases**
25/02-03/03	1	Interfaçage avec NMLP. Écriture des opérateurs de serialisation.
04/03-10/03	2	Utilisation d'une base de données pour activer la persistance des objets.
11/03-24/03	3..4	Écriture du superviseur. Utilisation des algorithmes de NMLP.
25/03-31/03	5	Test du superviseur sur un jeu de données simple.
01/04-05/05	**6..10**	**Phase 2 : Développements**
01/04-14/03	6..7	Mise en serveur autour d'un contrôleur.
15/04-28/03	8..9	Parallélisation du serveur.
29/04-05/05	10	Debug, test et documentation.
09/05	*11*	*Remise du rapport final.*
13/05-20/05	*12*	*Soutenance.*