Tâche 5 : Post-traitement et analyse statistique des résultats

EDF Lab Paris-Saclay – 2023-06-08

Pablo de Oliveira
<pablo.oliveira@uvsq.fr>

UNIVERSITÉ DE VERSAILLES ST-QUENTIN-EN-YVELINES

université PARIS-SACLAY

PROJET FINANCÉ PAR L'ANR - PROJECT FUNDED BY THE ANR
ANR

Objectifs et sous-tâches

- 5.1 Analyse et modèles statistiques
  - Modéliser la distribution des erreurs en arithmétique stochastique
- 5.2 Localisation d'erreurs
  - Localiser les erreurs numériques au sein d'un programme
- 5.3 Visualisation de données
  - Représentations graphiques pour interpréter les résultats produits par les outils d'Interflop

5.1 Analyse et modèles statistiques

- Estimateur de chiffres significatifs pour arith. stochastiques [TOMS'22]
  - Distributions normales et non-normales
  - Intégré à Vérificarlo et Verrou
  - Paquet Python *significantdigits* [Y. Chatelain, https://github.com/verificarlo/significantdigits]

- Analyse d'erreur pour l'arrondi stochastique [SISC'23]
  - Thèse de E.M. El-Arar (**présentation vendredi**)
  - Méthodes pour analyser des algorithmes : variance ou martingale
  - Bornes d'erreurs pour des algorihmes linéaires et non-linéaires
    - produit scalaire, schéma d'Horner, somme Pairwise, calcul de la variance, …

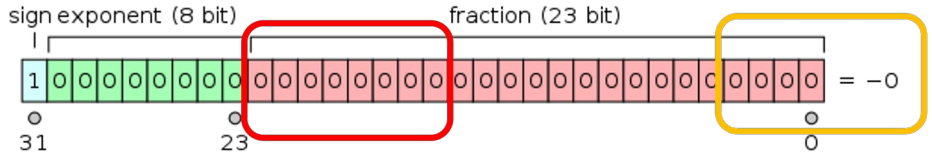
- Modèles d'arrondi stochastique non-déterministe, B. Lathuilière (**présentation vendredi**)

5.2 Localisation d’Erreurs

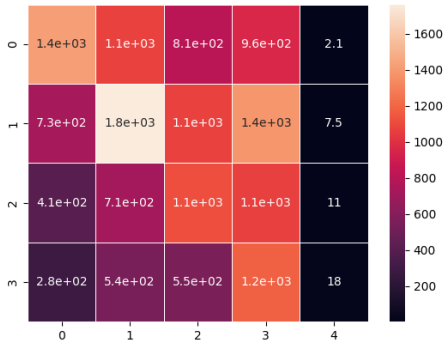
- Localisation au sein d’un code - Delta-Debug [Zeller]
  - Implémentation pour backends asynchrones [B. Lathuilière]
    → Intégrée à Verrou et Verificarlo
  - Implémentation synchrone
    → Promise
- Verificarlo-CI : suivi des versions de code pour intégration continue
  → Développé au sein du CoE TREX, intégré à Verificarlo

5.3 Visualisation : Analyse chromatique

- Objectifs
  - Suivi des valeurs numériques dans un programme
- Réalisation
  - Bibliothèque Python/C++ de surcharge d'opérateur
  - Associe une couleur à un scalaire ou groupe de scalaire (solution au pb du fléau de la dimension)
- Partenaires : CEA-UPVD



Chromatic analysis **Error analysis**



Application en algèbre linéaire

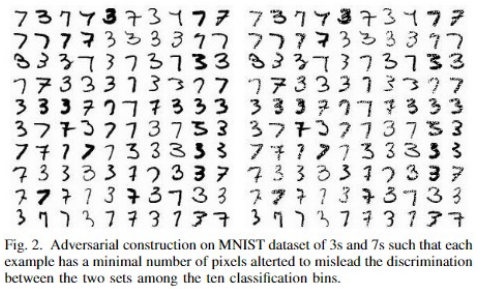
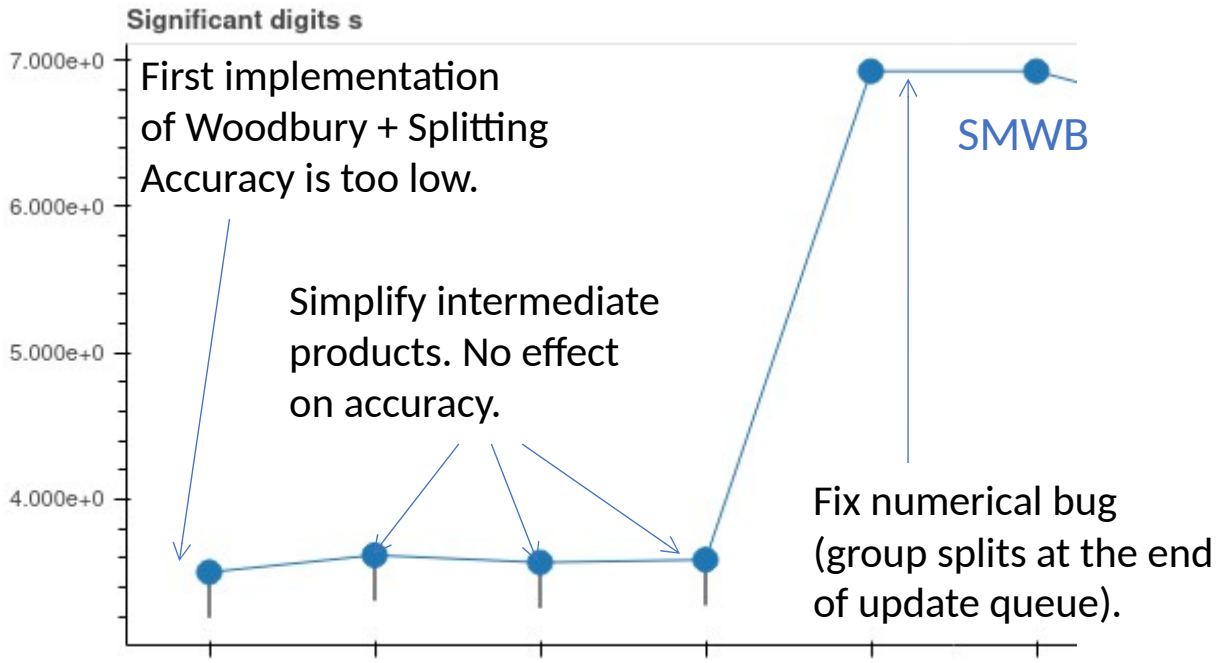


Fig. 2. Adversarial construction on MNIST dataset of 3s and 7s such that each example has a minimal number of pixels alterted to mislead the discrimination between the two sets among the ten classification bins.

Application en apprentissage

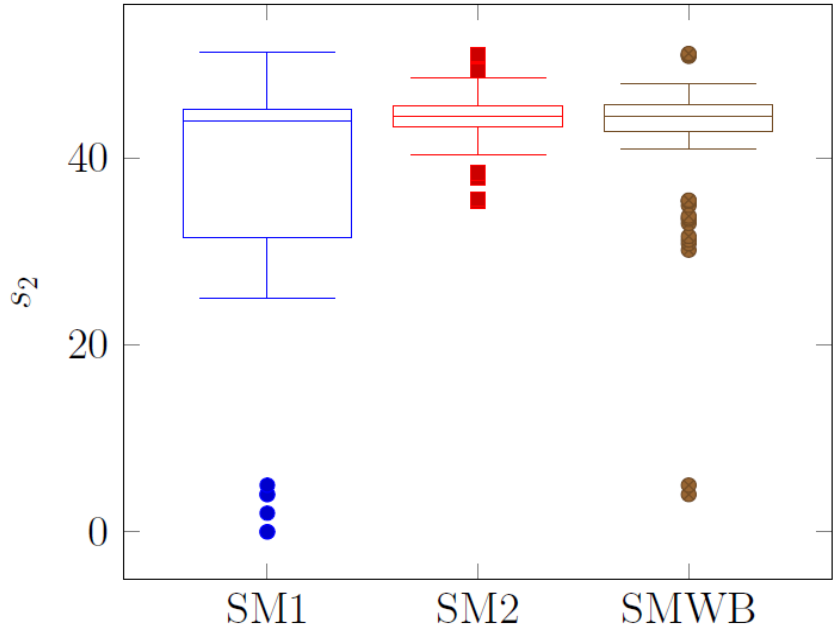
5.3 Visualisation : VFC-CI

Génération de rapports automatiques à chaque commit GIT.



Git commits (QMCkl – SMWB kernels)

Vues qui comparent différentes implémentations d'un même noyau.



Significant bits of three versions of the Sherman-Morrison kernel.

5.3 Visualisation : Outils de post-traitement

Outils intégrés à Verrou. Extensibles à d'autres frontends ?

- verrou_plot_stat
  - Génération automatique d'histogrammes
  - Même interface que l'interface ddebug

- post_verrou_dd : post-traiter des résultats de delta-debug avec
  - Un nombre différent de run
  - D'autres modes d'arrondi (format compatible avec verrou_plot_stat)
  - Avec une sélection des instructions (add,sub,....)
  - Avec la couverture par basic-bloc

Discussion

- Points oubliés / changements par rapport au projet initial ?
  - Localisation plus fine
  - Mutualisation des outils de post-traitement : delta-debug / visualisation ?

- Objectifs pour la dernière année ? Perspectives à plus long terme ?