Rapport de Travaux Pratiques 2

Amélioration de la Robustesse et de la Performance des Modèles

TOLOKOUM David 21P478
Département Génie Informatique (M2-GI)

Résumé

Ce rapport documente la réalisation du Travail Pratique 2, centré sur l'application de techniques avancées pour améliorer les performances d'un classifieur MLP sur MNIST. Nous avons abordé le diagnostic des erreurs (Biais/Variance), l'implémentation de la régularisation (L2, Dropout), la comparaison des optimiseurs (Adam, RMSprop, SGD+Momentum) et l'impact de la Normalisation par Lots (Batch Normalization). Les résultats sont tracés et analysés via la plateforme MLflow.

Table des matières

1 Partie 1 : Théorie et Concepts Clés

1.1 1.1 Diagnostic de Performance : Biais vs. Variance

1.1.1 Importance de la Séparation des Données (Training, Validation, Test)

Il est crucial de diviser le jeu de données en trois ensembles distincts pour garantir un diagnostic de performance fiable et une bonne généralisation :

1. **Ensemble d'Entraînement (*Training Set*)** : Utilisé pour apprendre les poids du modèle $(\mathbf{W}, \mathbf{b})$.
2. **Ensemble de Validation (*Validation ou Dev Set*)** : Utilisé pour l'ajustement des hyperparamètres (nombre de couches, taux d'apprentissage, taux de dropout, etc.). Il fournit une évaluation intermédiaire impartiale.
3. **Ensemble de Test (*Test Set*)** : Utilisé une seule fois, à la toute fin du projet, pour donner une estimation finale et non biaisée de la performance du modèle sur de nouvelles données.

1.1.2 Analyse des Erreurs pour le Diagnostic

En examinant l'erreur (ou la précision) sur l'ensemble d'entraînement et de validation, nous pouvons diagnostiquer le problème du modèle :

— **Biais Élevé (*High Bias* / Sous-apprentissage)** : L'erreur sur l'ensemble d'entraînement est **élevée** (Ex : Erreur Train $\approx 15\%$) et l'erreur de validation est également **élevée** et proche (Ex : Erreur Val $\approx 16\%$). Le modèle n'est pas assez complexe pour capturer les relations.

— **Variance Élevée (*High Variance* / Sur-apprentissage)** : L'erreur sur l'ensemble d'entraînement est **faible** (Ex : Erreur Train $\approx 1\%$) mais l'erreur de validation est **beaucoup plus élevée** (Ex : Erreur Val $\approx 10\%$). Le modèle a mémorisé le bruit des données d'entraînement et échoue sur de nouvelles données.

— **Bonne Performance (*Low Bias, Low Variance*)** : Les deux erreurs sont faibles et proches.

1.2 1.2 Régularisation et Normalisation

1.2.1 Régularisation L2 (*Weight Decay*)

Le principe de la régularisation L2 est d'ajouter un terme à la fonction de coût qui pénalise les poids $\mathbf{W}$ du modèle s'ils deviennent trop grands. La fonction de coût devient :

$$J_r(W, b) = J(W, b) + \frac{\lambda}{2m} \sum_{l=1}^{L} ||W^{[l]}||_F^2$$

Où λ est l'hyperparamètre de régularisation. En forçant les poids à rester petits, la fonction d'activation devient plus linéaire, ce qui rend la fonction de décision plus simple et moins susceptible de sur-apprentissage (réduit la Variance).

1.2.2 Dropout

Le Dropout fonctionne en désactivant aléatoirement (mettre à zéro) un pourcentage fixe de neurones dans les couches cachées lors de **chaque étape d'entraînement**.

— **Mécanisme** : Un neurone ne peut pas compter sur ses entrées pour être toujours présentes.

— **Régularisation** : Cela force le réseau à apprendre des représentations plus robustes et distribuées, car aucune caractéristique spécifique ne peut dominer la prédiction. Le Dropout peut être vu comme l'entraînement de nombreux sous-réseaux aléatoires partageant des poids. Il réduit la Variance.

1.2.3 Batch Normalization (BN)

La Normalisation par Lots standardise les activations des couches cachées, juste avant la fonction d'activation (ou après, selon les implémentations).

- **Mécanisme** : Pour chaque mini-lot, les activations intermédiaires sont normalisées pour avoir une moyenne $\mu = 0$ et une variance $\sigma^2 = 1$.
- **Stabilisation et Accélération** : La BN réduit le problème de la « covariable shift interne » (*Internal Covariate Shift*), assurant que la distribution des activations des couches cachées reste stable, même si les paramètres des couches précédentes changent. Cela permet d'utiliser des taux d'apprentissage plus élevés et **accélère considérablement la convergence**.

1.3 1.3 Algorithmes d'Optimisation Avancés

- **Momentum** : Ajoute un terme de "vitesse" (un moment exponentiellement pondéré des gradients précédents) à la mise à jour des poids. Il aide l'optimiseur à continuer à se déplacer dans une direction cohérente, amortissant les oscillations et accélérant la convergence dans les directions pertinentes.
- **RMSprop (*Root Mean Square Propagation*)** : Utilise une moyenne mobile exponentiellement pondérée des **carrés** des gradients passés pour adapter le taux d'apprentissage de chaque paramètre. Il diminue le taux d'apprentissage pour les dimensions avec de grands gradients (axes escarpés) et l'augmente pour les dimensions avec de petits gradients (axes plats).
- **Adam (*Adaptive Moment Estimation*)** : Combine les avantages de Momentum et de RMSprop. Il utilise à la fois les moments moyens (*mean*) des gradients (comme Momentum) et les moments carrés moyens (*variance*) des gradients (comme RMSprop).
- **Choix par Défaut** : Adam est souvent considéré comme l'algorithme par défaut car il est robuste, requiert peu de réglages d'hyperparamètres (le taux d'apprentissage par défaut fonctionne souvent bien) et converge généralement de manière rapide et fiable sur une grande variété de problèmes.

2 Partie 2 : Exercices Pratiques et Analyse

2.1 2.1 Exercice 1 : Diagnostic Biais/Variance

Le modèle de base du TP 1, entraîné avec les nouvelles données et une séparation Train/Dev/Test, a produit les métriques typiques suivantes après 5 époques (basé sur le run Adam initial) :

- **Précision Entraînement** : $\approx 98.4\%$
- **Précision Validation (Dev)** : $\approx 97.9\%$

Diagnostic :

- **Biais (Sous-apprentissage)** : L'erreur d'entraînement est faible (Biais faible), car 98.4% de précision est excellent pour un MLP sur MNIST. Le modèle est assez puissant.
- **Variance (Sur-apprentissage)** : Il existe un léger écart entre l'erreur d'entraînement ($\approx 1.6\%$) et l'erreur de validation ($\approx 2.1\%$). Cela indique une légère Variance Élevée. Le modèle sur-apprend légèrement l'ensemble d'entraînement.

Conclusion : Le modèle est globalement performant (faible biais) mais nécessite une régularisation pour réduire le sur-apprentissage (variance).

2.2 2.2 Exercice 2 : Application de la Régularisation (L2 et Dropout)

Suite au diagnostic de variance élevé, la **régularisation L2** (avec $\lambda = 0.001$) et une couche de **Dropout (0.2)** ont été ajoutées au modèle.

Analyse d'Impact : L'objectif de cette modification était de réduire la Variance. En pratique, l'ajout de ces régularisations (présentes dans les modèles des exercices suivants) a permis de stabiliser la performance et de réduire l'écart entre les métriques Train et Validation, bien que la précision de test maximale n'ait pas nécessairement augmenté de manière spectaculaire (puisque le modèle initial était déjà bon). L'ajout de régularisation est une mesure préventive contre l'overfitting.

2.3 2.3 Exercice 3 : Comparaison des Optimiseurs

Trois optimiseurs ont été comparés en utilisant la même architecture (régularisée) sur 10 époques. Les résultats de précision finale sur l'ensemble de test sont résumés dans le Tableau 1.

TABLE 1 – Comparaison des Précisions Finales des Optimiseurs (sur 10 époques)

Optimiseur	Précision Test Finale	Observation
Adam	0.9825	Très bonne performance, convergence rapide.
RMSprop	**0.9828**	Meilleure précision sur ce jeu d'hyperparamètres.
SGD + Momentum	0.9726	Plus lent à converger.

Analyse :

— RMSprop a légèrement surpassé Adam, démontrant son efficacité sur les problèmes de classification d'images.

— Adam a montré une excellente performance. En examinant les courbes MLflow, Adam et RMSprop affichent une descente rapide et stable de la Loss.

— SGD + Momentum a été le moins performant sur 10 époques, nécessitant typiquement plus de temps pour atteindre la convergence optimale par rapport aux optimiseurs adaptatifs.

2.4 2.4 Exercice 4 : Impact de la Batch Normalization (BN)

L'algorithme Adam a été utilisé pour tester le modèle avec l'ajout de la couche `BatchNormalization` après la première couche Dense.

Comparaison des Résultats (Adam vs. Adam + BN) sur 10 époques :

— Vitesse et Stabilité : L'observation des logs de l'entraînement (Epoch 1/10 à Epoch 10/10) montre que le modèle avec BN atteint une haute précision de validation plus tôt (meilleure performance dès l'époque 3-4) et que les métriques d'entraînement sont beaucoup plus lisses. La BN a réussi à accélérer et stabiliser le processus.

— Performance Finale : La précision finale sur l'ensemble de test (0.9791) est légèrement inférieure à celle du meilleur run sans BN (0.9828 avec RMSprop). Cependant, cette légère baisse est souvent compensée par la rapidité de convergence. Si l'entraînement était poursuivi avec un nombre d'époques plus élevé, le modèle avec BN serait plus susceptible de maintenir une haute performance.

3 Conclusion

Ce Travail Pratique a permis de transformer un modèle fonctionnel (TP 1) en un modèle plus robuste et mieux optimisé. Nous avons confirmé l'importance des outils de diagnostic (Biais/Variance), l'efficacité de la régularisation (L2 et Dropout) pour le contrôle de la variance, et les gains de performance et de stabilité apportés par les optimiseurs adaptatifs (Adam/RMSprop) et la Normalisation par Lots (Batch Normalization).