Membres du Groupe :

- Lilia Izri
- Anis Smati

Rapport de Projet

Apprentissage supervisé

Introduction :

Le but du projet est l'application de l'algorithme du Perceptron Multi-classes sur un Jeu de Données composé des lettres de l'Alphabet Latin en Majuscule. Dans un premier temps, on fera varier certains paramètres et méta-paramètres de l'algorithme afin de comparer les différents scores et ainsi obtenir les meilleures performances possibles. Puis, dans un second temps, On illustrera Les résultats obtenus dans ce rapport.

Structure:

Le Projet est divisé en trois classes :

- **La Classe ImagePerceptronMultiProject :** Contient le chemin de récupération des données ***'path'*** **à indiquer !** ainsi que les '***constantes'*** et les '***méthodes***' nécessaires pour l'initialisation du jeu de données (i.e. Ensemble d'apprentissage, de validation, de tests ainsi que les poids des Perceptrons).
- **La Classe FonctionsResultatsProject :** Contient La majorité des '***méthodes***' Permettant l'obtention des résultats :
  - La Matrice de Confusion.
  - L'allure Moyenne des classes inférées par le perceptron.
  - Les 5 images bien classées avec le plus bas score d'inférence.
  - Les images mal classées avec le plus bas score d'inférence d'une classe donnée en paramètre.
  - Plus quelques méthodes pour faciliter la réalisation des méthodes citées plus haut.
- **La Classe PerceptronMultiProject :** Contient L'algorithme '***Principal'*** du Perceptron ainsi que Les '***méta-paramètres'*** et la fonction ***'Main***' permettant d'obtenir les Résultats demandés.

Résultats :

En exécutant la classe **PerceptronMultiProject,** On initialise les données à l'aide de la fonction '*initialisationDonnees*' On entraine un perceptron sur ces données. Un dossier 'resultats' se crée automatiquement on y notera les résultats numériques dans des fichiers CSV et on y enregistrera les illustrations sous forme d'images png.

Q1/ **Objectif :** En fixant Nv=1000, et en faisant varier Na de 1000 à 10000. Trouver le Meilleur Na.

-**Démarche :** On Entraine le perceptron avec eta=0.001 et au fur et à mesure des époques on calculera le nombre d'erreurs sur l'ens d'apprentissage à l'aide de la fonction 'epoque' et le nombre d'erreurs sur l'ens de validation à l'aide de la fonction 'nbErreur' et on notera les résultats dans le fichier csv « Resultats_nbErreur_k=12.csv». On recommencera en modifiant la valeur de Na dans la classe ImagePerceptronMultiProject.

-**Observation :** On remarque que le nombre d'erreurs diminue considérablement quel que soit les valeurs de Na avant de se stabiliser. Cependant, on remarque que le nombre d'erreurs est moindre avec Na=10.000 et Na=7000. On obtiendra cependant de meilleurs scores sur test avec Na=10.000.

-**Conclusion :** On choisit donc Na=10.000.

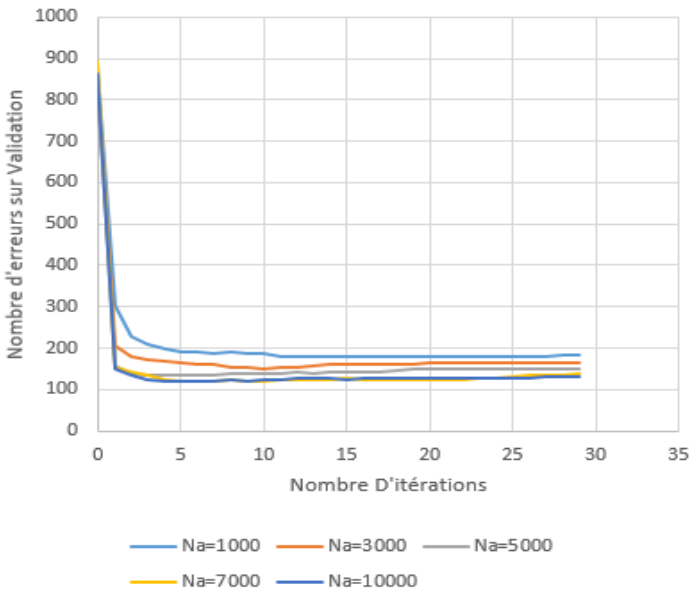


Figure 1: Graphe représentant le nombre d'erreurs sur Validation en fonction Des valeurs de Na.

Q2/ **Objectif :** En fixant Nv=1000, Na=10.000, Faire Varier le taux d'apprentissage pour choisir le meilleur Eta, puis Calculer le score final sur l'ensemble de test.

-**Démarche :** On Entraine le perceptron et au fur et à mesure des époques on calculera le nombre d'erreurs sur l'ens d'apprentissage à l'aide de la fonction '*epoque*' et le nombre d'erreurs sur l'ens de validation à l'aide de la fonction '*nbErreur*', on calculera également le score Finale sur l'ensemble de Test toujours à l'aide de la fonction '*nbErreur*' et on notera tous les résultats dans le même fichier que la question 1. Et on recommencera en modifiant la valeur de eta dans La classe PerceptronMultiProject.

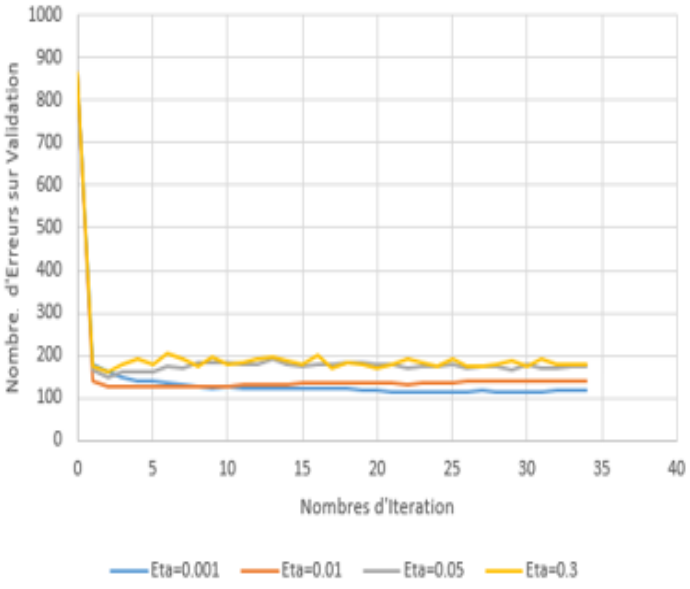


Figure 2: Graphe représentant le nombre d'erreurs sur Validation en fonction des valeurs de eta.

-Observation : On remarque une fois encore que le nombre d'erreurs diminue considérablement avec ces valeurs de 'eta' avant de se stabiliser plus ou moins. On remarque également que le nombre d'erreurs est moindre avec eta= 0.001 et eta=0.005. Cependant, au bout de 15 itérations on remarque qu'avec eta=0.001 on obtient moins d'erreurs.

-Conclusion : - On choisit donc eta=0.001.

- **SCORE** sur Test : 109 Erreurs /1000 données (≈11% d'erreurs).

-**Remarques /!\ : -** En prenant un eta > 0.4, On obtient des NaN dans InfPerceptron ce qui fausse tous les résultats, tout comme, si on prend un eta trop en dessous de 0.001, l'algorithme ne converge jamais c'est pour cela qu'on se restreindra à eta= [0.001, 0.3].

- On fixera epochmax à 30-35 car au-delà le nombre d'erreurs sur l'ens de validation commence à augmenter doucement.

-N.b : Les fonctions '*Epoque*' et '*nbErreur*' sont dans la classe principale. Toutes les autres fonctions citées en dessous sont dans la classe FonctionsResultatsProject.

Q3/ -**Objectif :** Pour Na=5000 et Nv=1000, Calculer la Matrice de Confusion puis commenter les classes problématiques.

-Démarche : On crée la matrice de confusion à l'aide de la fonction '*matriceConfusion*' De la classe FonctionsResultatsProject. Puis on enregistrera les valeurs dans le fichier csv « Matrice_De_Confusion.csv ».

	Vrai A	Vrai B	Vrai C	Vrai D	Vrai E	Vrai F	Vrai G	Vrai H	Vrai I	Vrai J	Vrai K	Vrai L
A	70.0	0.0	1.0	0.0	0.0	3.0	0.0	3.0	0.0	0.0	2.0	0.0
B	0.0	55.0	1.0	0.0	2.0	1.0	3.0	0.0	0.0	0.0	0.0	0.0
C	1.0	2.0	162.0	0.0	2.0	3.0	4.0	0.0	0.0	2.0	2.0	3.0
D	0.0	3.0	0.0	67.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	1.0
E	1.0	2.0	0.0	1.0	62.0	1.0	3.0	0.0	1.0	0.0	4.0	0.0
F	3.0	0.0	1.0	0.0	5.0	116.0	0.0	0.0	1.0	0.0	2.0	0.0
G	1.0	0.0	0.0	0.0	1.0	3.0	26.0	1.0	1.0	0.0	0.0	0.0
H	7.0	2.0	0.0	0.0	0.0	1.0	1.0	34.0	0.0	0.0	0.0	0.0
I	0.0	0.0	0.0	1.0	1.0	3.0	0.0	0.0	150.0	4.0	1.0	3.0
J	0.0	1.0	2.0	3.0	0.0	1.0	0.0	0.0	3.0	46.0	0.0	1.0
K	1.0	0.0	2.0	0.0	0.0	1.0	0.0	0.0	1.0	0.0	29.0	1.0
L	0.0	1.0	4.0	2.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	62.0

Figure 3: Matrice de Confusion.

Classes problématiques : On remarque souvent que ce sont les Classes des lettres qui ont le plus de 'points communs' (Qui ont une forme ou une partie assez ressemblante) qui se révèlent problématiques, par exemple :

- 7 images de la lettre A sont confondus avec la lettre H.
- 5 images de la lettre E sont confondues avec la lettre F.
- 4 images de la lettre C sont confondues avec la lettre L.
- Presque un tiers des images de la lettre K sont confondues avec les lettres E, A et B.
- Presque un tiers des images de la lettre G sont confondues avec les lettres C, E et B.

Q4/-**Objectif :** Trouver l'allure Moyenne de chaque Classe Inférée par le Perceptron.

-Démarche : Pour chaque classe : On récupère d'abord les images bien classées à l'aide de la fonction '*regroupeImagesBienClassees*' (on connait leur nombre grâce à la matrice de confusion) puis on calcule L'image moyenne à l'aide de la fonction '*imageMoyenne*' Et on enregistre l'image Dans le dossier « resultats/Illustrations » à l'aide de la fonction 'sauvegardeImg'. Sous forme : « ImgMoyenne_De_La_Classe_'numero de la classe'.png»

N.b : Toutes ces fonctions sont dans la classe FonctionsResultatsProject dans la partie 'Fonctions pour resultats'.

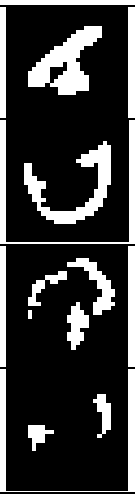
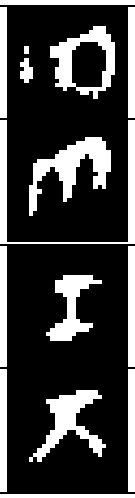
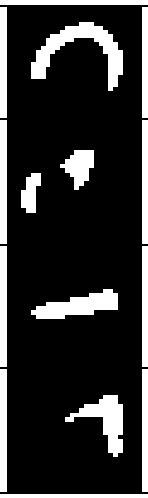
Classe A :		**Classe B :**		**Classe C :**	
Classe D :		**Classe E :**		**Classe F :**	
Classe G :		**Classe H :**		**Classe I :**	
Classe J :		**Classe K :**		**Classe L :**	

Figure 4: Illustration de l'allure moyenne de chaque classe inférée par le perceptron.

Q5/-Objectif : Illustrer 5 images bien classées avec le plus mauvais score d'inférence.

-Démarche : Pour récupérer les images bien classées de l'ensemble de validation on doit connaitre d'abord leur nombre, pour cela on somme la diagonale de la matrice de confusion. Ceci fait, on récupère les indices (dans l'ens de validation) et le taux d'inférence des images bien classées à l'aide de la fonction '*bienClasseesAll*'. Puis on « extrait » les 5 avec le taux d'inférence le plus bas à l'aide de la fonction '*minInference*'. Et finalement, on les sauvegarde l'aide de la fonction '*sauvegardeImg*' sous forme : « Imgi_BienClassee_De_La_Classe_'numero de la classe'.png» Et on écrira leur taux d'inférence dans un fichier csv « resultats_inferences_images.csv »

-Commentaire : Lorsque l'on regarde chaque image séparément (cf. figure 5) on voit qu'elle ne ressemble pas tellement à l'image moyenne inférée par le perceptron pour la classe de cette image. Mais apparemment suffisamment pour être bien classée.

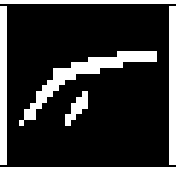
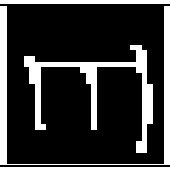
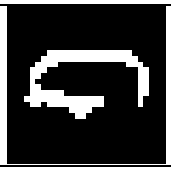
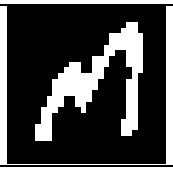
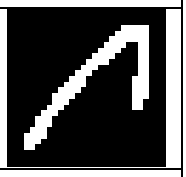
	Image 1 **Classe F :**	**Image 2** **Classe E :**	**Image 3** **Classe C :**	**Image 4** **Classe E :**	**Image 5** **Classe L :**
Image					
Taux d'inf	0.249429	0.2684394	0.3192479	0.3324305	0.3527737

Figure 5: Illustration de 5 images bien classées avec le taux d'inférence le plus bas.

Q6/-**Objectif :** Illustrer sur 5 classes (ici les classes de F à J) les Images mal classées avec le pire score d'inférence.

-Démarche : Pour chaque classe : Pour récupérer le nombre d'images mal classées d'une certaine classe de l'ensemble de validation on doit connaitre d'abord leur nombre, pour cela on somme la colonne correspondant à la classe dans la matrice de confusion et on soustrait le nombre d'images bien classées. Comme pour la question précédente, on récupère les indices et les taux d'inférence à l'aide de la fonction '*malClasseeC*' et on 'extrait' celle avec le plus mauvais taux d'inférence à l'aide de la fonction '*minInference*'. Et finalement, on les sauvegarde à l'aide de la fonction '*sauvegardeImg*' sous forme : « Imgi_MalClassee_De_La_Classe_'numéro de la classe'.png» Et on écrira leur taux d'inférence dans Le même fichier précèdent « resultats_inferences_images.csv ».

-Commentaire : Lorsque l'on regarde L'image de chaque classe (cf. figure 6) on voit bien qu'on est loin de l'image moyenne inférée par le perceptron pour cette même classe, il serait donc normal qu'elle soit mal classée. (En même temps, même nous on a du mal à les différencier celles-ci).

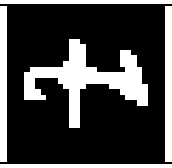
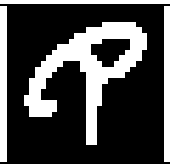
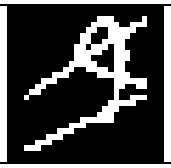
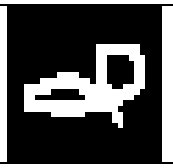
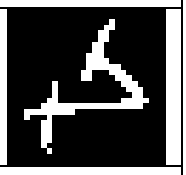
	Classe F :	**Classe G :**	**Classe H :**	**Classe I :**	**Classe J :**
Image					
Taux d'inf	0.00126	0.0123813	0.00277	0.018121516	0.0199593

Figure 6: Illustration de l'image la plus mal classée des classes F-J

Q7/-**Objectif :** Tracer la courbe d'erreurs sur L'ensemble d'apprentissage et de validation en fonction du nombre d'Itérations, avec les 26 classes (Toutes les lettres majuscules), Puis donner le score.

-Démarche : Comme pour la question 1 ou 2, on réinitialise les données (à fin de réinitialiser w) et on entraine un nouveau perceptron avec les 26 Lettres, avec epochmax à 20. Et on notera les résultats dans le fichier csv « Resultats_nbErreur_k=26.csv».

-SCORE sur Test : 198 Erreurs /1000 données (≈20% d'erreurs).

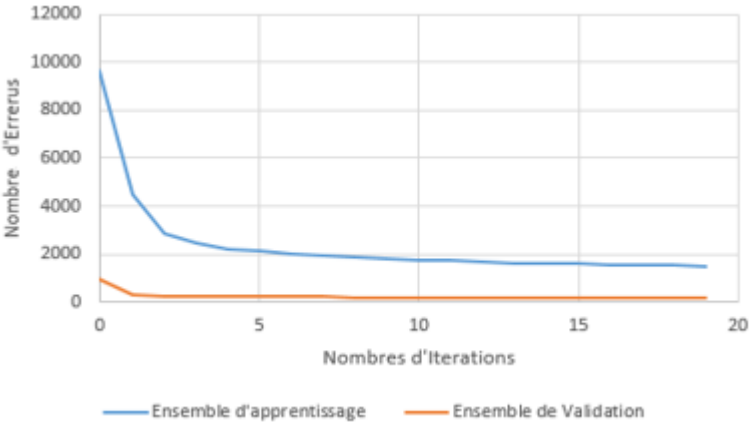


Figure 7: Graphe représentant le nombre d'erreurs sur l'ens d'apprentissage et de validation pour K=26

Conclusion : L'algorithme du perceptron multi classes semble être un « assez-bon » modèle sur ce jeu de données. On arrive à avoir d'assez bons résultats en un temps pas très grand. On pourrait également peut-être avoir de meilleurs scores en jouant sur les paramètres ou autre.
-Pas de 'vrais' problèmes rencontrés.
Yes ! 4 pages pile-poil !