Rapport de projet : What the hell is that

Nom du groupe : Camera (vision)
Nom du groupe administratif : 3
Membres de l'équipe :
Alexandre Bonin – Damien Merret – Alexandre l'Heritier – Kenyu Kobayashi – Anass El Anbari
Lien de la competition : http://codalab.lri.fr/competitions/111/
Numéro de la dernière soumission Codalab : 4550
Lien video : https://www.youtube.com/watch?v=VSEfF5iVO0w&feature=youtu.be
Lien github : https://github.com/kenkob09/CameraProject
Lien du diapo : https://upsud-my.sharepoint.com/:p:/g/personal/alexandre_lheritier_u-psud_fr/Edi9GelYmMRIttjkZsX-VywBGSzQ6uJdBgXO9GpzkspAEg?e=nZn43I

Introduction

"What the hell is that" est un projet dont l'objectif est de classifier des images abstraites en des vecteurs de dimension 256 en 10 classes différentes. Le participant au projet reçoit trois ensembles de données différents, L'ensemble de données d'entraînement contient 40 000 exemple, celui de validation et de test en contiennent 10 000 chacun. Les ensembles de données sont équilibrés. Chaque composante d'un vecteur d'entrée est une caractéristique mathématique abstraite et ne sera pas détaillée ici. Le projet fournit un classifieur de base obtenant un score de 87% après apprentissage sur l'ensemble d'entraînement. L'objectif du projet est d'ajuster les hyper paramètres du modèle afin de maximiser le score.

Descriptions des algorithmes étudiés et pseudo-code

1) *Preprocessing : Anass El Anbari*

Le Preprocessing sert à filtrer les données que le modèle prédictif utilisera pour l'apprentissage. J'ai effectué ceci en utilisant l'algorithme suivant: En créant dans un premier temps un objet de VarianceThreshold, j'élimine les données avec une influence faible sur l'apprentissage. Le seuil que j'ai choisis pour VarianceThreshold était 4 suite aux résultats préliminaires (Figure 1). J'ai choisis VarianceThreshold comme méthode finale car le code du modèle prédictif a besoin de résultats positifs, et après avoir renvoyé les valeurs absolues de mon preprocessing dans les 3 méthodes principales –fit, fit_transform, transform-. la précision du modèle chute brutalement si j'utilise PCA sur 2 dimensions, StandardScaler, ou MinMaxScaler; De nombreuses combinaisons de ces 3 méthodes avec VarianceThreshold dans une pipeline donnent toujours une précision médiocre même en renvoyant des données positives et négatives. Mais en n'utilisant que VarianceThreshold, la précision du modèle prédictif s'améliore considérablement.

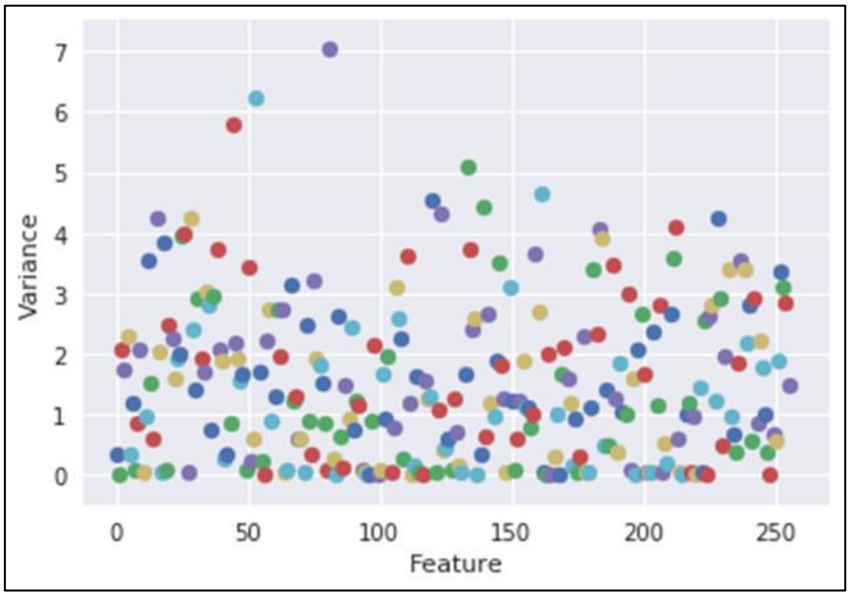

Figure 1 : Variance versus numéro de la caractéristique.

On observe ici un très nombre de features non significatrices. On peut d'ores et déjà fixer un seuil minimal de variance, par exemple 4, pour supprimer les données qui ne nous aideront pas.

2) *Modèle predictif : Kenyu Kobayashi, Damien Merret*

Le choix du ***modèle prédictif*** est trivial pour une bonne classification des données.

On a donc débuté par estimer la performance de plusieurs modèles prédictifs à notre disposition pour trouver celui qui correspond le mieux à notre base de données.

Cette estimation s'est faite grâce à une procédure de validation croisée, le k-fold.

Elle fonctionne de la manière suivante : On divise le jeu de données en k échantillons, puis on sélectionne un des k échantillons qui servira d'ensemble de test. Les (k-1) à autres échantillons constitueront l'ensemble d'apprentissage. Ensuite, on estime la performance du modèle de la même manière qu'avec la première méthode. On répète la même opération avec un autre échantillon comme ensemble de test parmi les (k-1) autres échantillons qui n'ont pas encore été utilisés pour valider le modèle. Ainsi, l'opération est répétée k fois, puis la performance du modèle est finalement estimée en faisant la moyenne des k estimations.

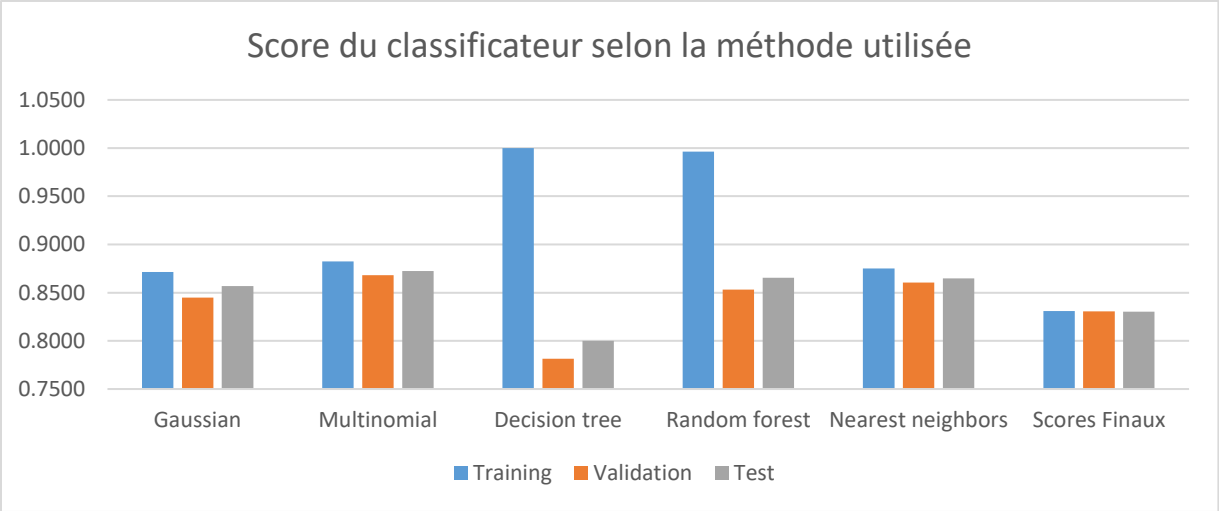

Figure 2 : Score du classificateur en fonction de la méthode utilisée – Histogramme.

Dataset	Gaussian	Multinomial	Decision tree	Random forest	Nearest neighbors	Scores Finaux
Training	0,8715	0,8823	1,0000	0,9962	0,8751	0,8308
Validation	0,8449	0,8682	0,7813	0,8531	0,8604	0,8305
Test	0,8568	0,8726	0,8000	0,8653	0,8648	0,8304

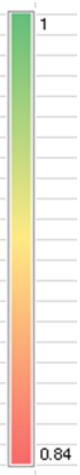

Figure 3 : score du classificateur en fonction de la méthode utilisée – Heat map.

On peut voir sur ce tableau que le meilleur score de validation croisée a été obtenue avec le classifieur Multinomial. C'est donc celui-ci qu'on a utilisé par la suite.

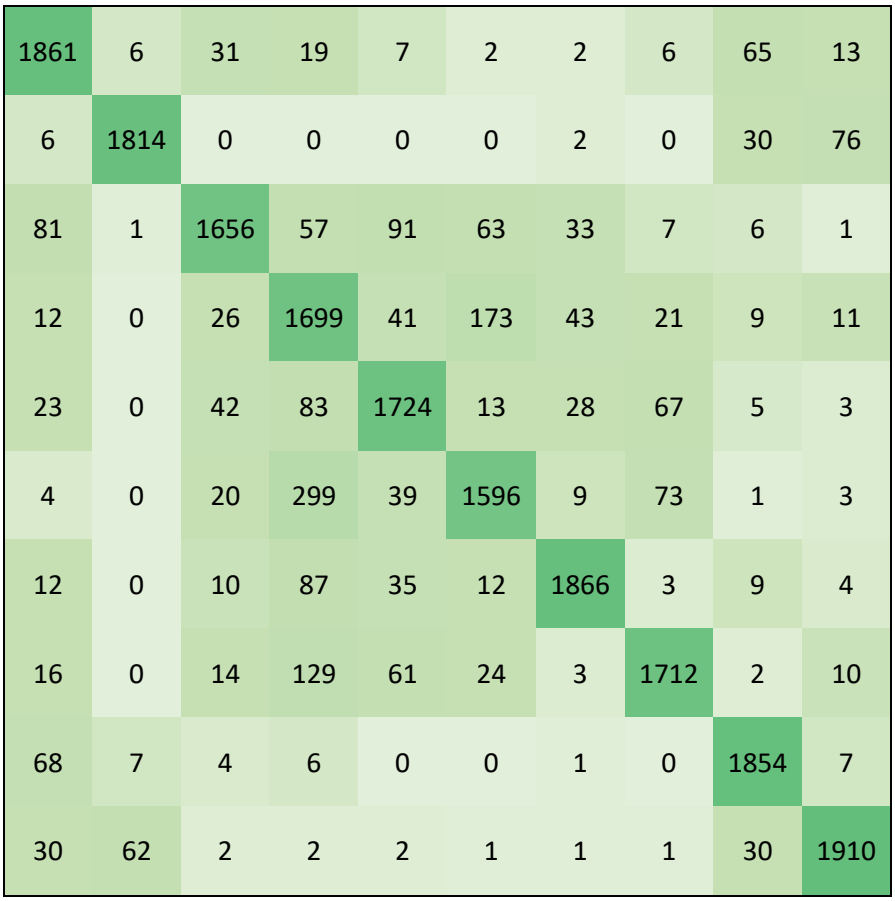

Figure 4 : Matrice de confusion du modèle MultinomialNB.

Après avoir trouvé le meilleur modèle grâce à la validation croisée, il fallait trouver les valeurs optimales pour les hyper paramètres de celui-ci. Cette recherche a pu faire par une procédure de Grid Search, un algorithme testant les performances du modèle pour plusieurs valeurs d'hyper paramètres et retournant les meilleures valeurs pour un score maximal.

Voici ce qu'il nous affiche en sortie :

```
Best parameters set found on development set:
        {'alpha': 0.05, 'fit_prior': False, 'class_prior': None}
Grid scores on development set:
        0.884 (+/-0.009) for {'alpha': 1, 'fit_prior': True, 'class_prior': None}
        0.885 (+/-0.008) for {'alpha': 0.1, 'fit_prior': True, 'class_prior': None}
        0.885 (+/-0.009) for {'alpha': 0.05, 'fit_prior': True, 'class_prior': None}
        etc..
```

3) *Visualisation : Alexandre Bonin, Alexandre l'Heritier*

La visualisation des données et des prédictions en un graphe à 2 dimensions est impossible sans traiter les données auparavant, puisqu'elles possèdent 256 dimensions. Par conséquent nous utilisons des algorithmes de réduction de dimension pour permettre leur visualisation. Les algorithmes utilisés sont TSNE et Isomap. Deux algorithmes différents sont utilisés pour permettre de comparer différentes représentations, la réduction de dimension perdant énormément d'informations sur les données originales. Voici un exemple de figure obtenu après apprentissage sur l'ensemble d'entraînement :

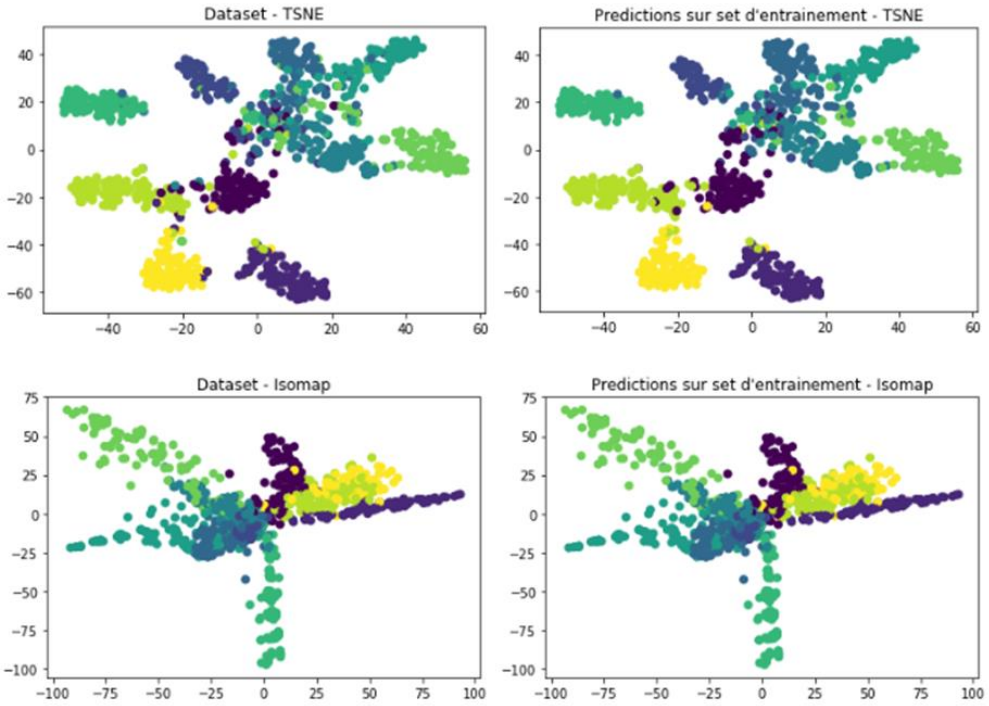

Figure 4 : Prédictions (à droite) des classes des données contre la vérité terrain (gauche), TSNE et Isomap.

La réduction de dimension par TSNE[1] est telle que deux points proches sur le graphe à 2 dimensions représentent deux données proches en 256 dimensions. Isomap[2] utilise des algorithmes de recherches de graphes (Dijkstra[3] et Floyd-Warshall[4]) et les distances euclidiennes pour construire une représentation à 2 dimensions. Les deux algorithmes nous permettent de visualiser les prédictions du modèle sur l'espace des données.

Cependant, ces algorithmes demandent un temps de calcul conséquent, ce qui peut entraver la tâche du modèle prédictif, à savoir ajuster les paramètres et recommencer un entraînement. Cela est encore plus ressenti sur l'ensemble d'entraînement, les 40 000 données sont une quantité trop importante d'information pour permettre au programme de visualisation de terminer. Nous avons donc décider de limiter le nombre de points affichés à 2000, étant le nombre d'exemples sur l'ensemble d'entraînement échantillon. Les 2000 données sont donc prises au hasard équitablement dans l'ensemble d'entraînement.

Annexe : Techniques et problèmes divers de machine learning

Dataset	Nombre d'exemples	Nombre de features	A des variables catégoriques ?	A des données manquantes	Nombre d'exemple pour chaque classe
Training	40 000	256	Non	Non	4000
Validation	10 000	256	Non	Non	1000
Test	10 000	256	Non	Non	1000

Figure 5 : Tableau descriptif des données du projets

Méthode	Gaussian classifier	Multinomial classifier	Decision Tree	Random Forest	Nearest Neighbors	Multinomial classifier + Preprocessing
Training	0.8715	0.8823	1.0	0.9962	0.8751	0.8308
CV	0.8449	0.8682	0.7813	0.8531	0.8604	0.8304
Valid(ation)	0.8568	0.8726	0.8000	0.8653	0.8648	0.8305

Figure 6 : Résultats préliminaires et résultat final (à droite)

Métrique et méthodes:

La métrique utilisée par notre challenge est le "**bac multiclass**" qui retourne bac_metric(solution, prediction, task='multiclass.classification')

Le code de bac_metric est

```python
def bac_metric(solution, prediction, task='binary.classification'):
    ''' Compute the normalized balanced accuracy. The binarization and
    the normalization differ for the multi-label and multi-class case. '''
    label_num = solution.shape[1]
    score = np.zeros(label_num)
    bin_prediction = binarize_predictions(prediction, task)
    [tn, fp, tp, fn] = acc_stat(solution, bin_prediction)
    # Bounding to avoid division by 0
    eps = 1e-15
    tp = sp.maximum(eps, tp)
    pos_num = sp.maximum(eps, tp + fn)
    tpr = tp / pos_num  # true positive rate (sensitivity)
    if (task != 'multiclass.classification') or (label_num == 1):
        tn = sp.maximum(eps, tn)
        neg_num = sp.maximum(eps, tn + fp)
        tnr = tn / neg_num  # true negative rate (specificity)
        bac = 0.5 * (tpr + tnr)
        base_bac = 0.5  # random predictions for binary case
    else:
        bac = tpr
        base_bac = 1. / label_num  # random predictions for multiclass case
    bac = mvmean(bac)  # average over all classes
    # Normalize: 0 for random, 1 for perfect
    score = (bac - base_bac) / sp.maximum(eps, (1 - base_bac))
    return score
```

- La méthode **Multinomiale NB** est utilisée pour classifier des données avec des caractéristiques discrètes
- L**'arbre de décision** est une méthode d'apprentissage qui créer un arbre avec pour nœuds, des décisions, et pour feuilles, des résultats après analyse de données étiquetées. Il peut ensuite soumettre une donnée inconnue à cette suite de décisions afin de fournir un résultat.
- La méthode **Random forest** consiste à utiliser plusieurs arbres de décisions sur divers échantillons de l'ensemble de données et à effectuer une moyenne sur les différents résultats.
- La méthode **Nearest Neighbors** consiste à trouver un nombre prédéfini d'échantillons d'entraînement le plus proche du nouveau point et de prédire l'étiquette à partir de ces points. Cet algorithme essaye donc d'attribuer à un point, l'étiquette des points les plus proches. Cette méthode peut être utilisé pour des problèmes de régression ainsi que de classification.

Le sur-apprentissage :

On appelle sur-apprentissage un phénomène commun aux modèles d'apprentissages. Tandis que le score de précision du modèle sur l'ensemble d'entraînement peut atteindre les 100%, passé un certain seuil variable selon le type de modèle et les données, on observe une perte de performances sur les ensembles jusqu'alors inconnus par le modèle.

On observe trois zones distinctes sur ce graphique. Tout d'abord la situation initiale, peu d'entraînement entraîne un score médiocre sur tous les ensembles (1/N où N est le nombre de classes). Cette zone est spécifique d'un « sous-entraînement », ou « sous-apprentissage ». Au contraire, on observe un « sur-apprentissage » par un éloignement des courbes d'erreur sur les deux ensembles. L'objectif est donc d'atteindre la zone idéale pour la spécification du modèle, situé au centre du graphe.

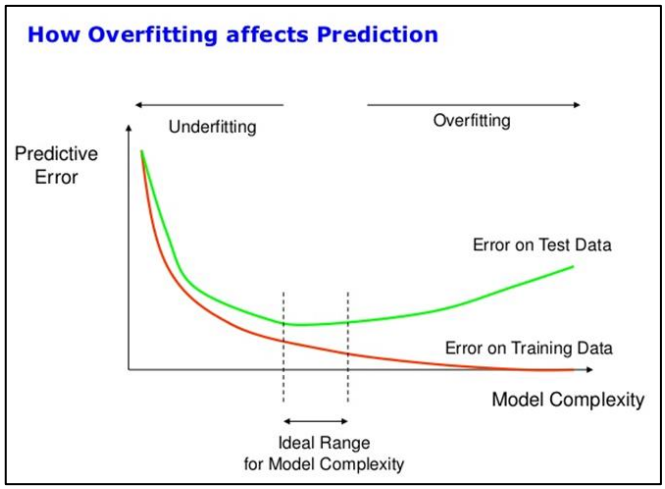

Figure 7 : Taux d'erreur versus spécification du modèle sur l'ensemble d'entraînement [12]

La cross-validation :

La méthode de cross-validation consiste à séparer les données en deux ou plusieurs sous-ensembles. L'un sera utilisé comme ensemble d'apprentissage afin de modifier les poids du réseau, l'autre deviendra l'ensemble de validation pour évaluer l'efficacité du réseau sur un ensemble de données inconnues.

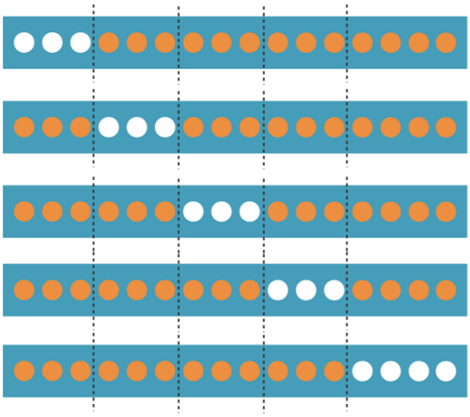

Figure 8 : Exemple de section d'un échantillon d'entraînement pour une validation croisée [13]

On répète l'opération sur la totalité de l'ensemble par sections successives pour éviter un éventuel « mauvais découpage » des données.

Bibliographie:

[1] van der Maaten, L.J.P.; Hinton, G.E. (Nov 2008). "Visualizing High-Dimensional Data Using t-SNE". Journal of Machine Learning Research. 9: 2579–2605

[2] J. B. Tenenbaum, V. de Silva, J. C. Langford, A Global Geometric Framework for Nonlinear Dimensionality Reduction, Science 290, (2000), 2319–2323.

[3] Cormen, Thomas H. (2001). "Section 24.3: Dijkstra's algorithm". Introduction to Algorithms (Second ed.). MIT Press and McGraw–Hill. pp. 595–601.

[4] Cormen, Thomas H. (2001). "The Floyd–Warshall algorithm". Introduction to Algorithms (Second ed.). MIT Press and McGraw–Hill. pp. 558-565.

[5] Scikit-learn : Model selection : http://scikit-learn.org/stable/tutorial/statistical_inference/model_selection.html

[6] Scikit-learn : Grid search : http://scikit-learn.org/stable/modules/grid_search.html

[7] Kdnuggets : Auto-Sklearn: https://www.kdnuggets.com/2016/08/winning-automl-challenge-auto-sklearn.html

[8] TSNE algorithm scikit-learn: http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

[9] Isomap algorithm scikit-learn: http://scikit-learn.org/stable/modules/generated/sklearn.manifold.Isomap.html

[10] Sklearn.feature_selection : VarianceThreshold :

http://scikit-learn.org/stable/modules/feature_selection.html

[11] Unsupervised dimensionality reduction's PCA :

http://scikit-learn.org/stable/modules/unsupervised_reduction.html

[12] Digvijay Singh (2012). “Overfitting & Transformation Based Learning” https://www.slideshare.net/ssrdigvijay88/overfitting-andtbl

[13] Chloée-Agathe Azencott. “Évaluez et améliorez les performances d’un modèle de machine learning” https://openclassrooms.com/courses/evaluez-et-ameliorez-les-performances-d-un-modele-de-machine-learning