Rapport de la Compétition 2
Crop Land Detection from Remote Sensing Data

Weiyue Cai
Département d'informatique et
de recherche opérationnelle
Université de Montréal
weiyue.cai@umontreal.ca

Steve Levesque
Département d'informatique et
de recherche opérationnelle
Université de Montréal
steve.levesque@umontreal.ca

Abstract

Le but de ce travail de compétition Kaggle est de classifier chaque point de données en 2 catégories: Non-crop land (0) ou Crop land (1). L'ensemble du jeu de données se compose de données mensuelles agrégées de télédétection (satellite), météorologie et topographie.

1 Algorithmes

1.1 KNN

(1) données originales, n_neighbours = 1, métrique par défaut (L2).
public score: 0.99757, private score: 0.99669 (meilleur résultat parmis les deux submissions finales)
(2) données originales, n_neighbours = 1, métrique: canberra.
public score: 0.99516, private score: 0.99669
(3) données originales, n_neighbours = 9, métrique: canberra.
public score: 0.90337, private score: 0.89318
(4) données originales, n_neighbours = 1, algorithm: ball tree, métrique: canberra.
public score: 0.99273, private score: 0.99779 (notre meilleur résultat pas sélectionné)
(5) données originales, n_neighbours = 1, algorithm: ball tree, métrique: chebyshev.
public score: 0.99514, private score: 0.99339

1.2 Voting

1.2.1 Hard voting

Avec les données originales et les algorithmes suivantes:

```
adaboost = AdaBoostClassifier(n_estimators=150)
knn = KNeighborsClassifier(n_neighbors=1)
lgbm = LGBMClassifier()
rf = RandomForestClassifier(n_estimators=350, max_depth=45, max_features='auto',
bootstrap=True, min_samples_leaf=1, min_samples_split=2)
xgboost = XGBClassifier(max_depth= 6, n_estimators=157,
learning_rate= 0.23989473738149508,
gamma= 0.7442032316202452, random_state=42)
```

public score: 0.98321, private score: 0.97396

35th Conference on Neural Information Processing Systems (NeurIPS 2021).

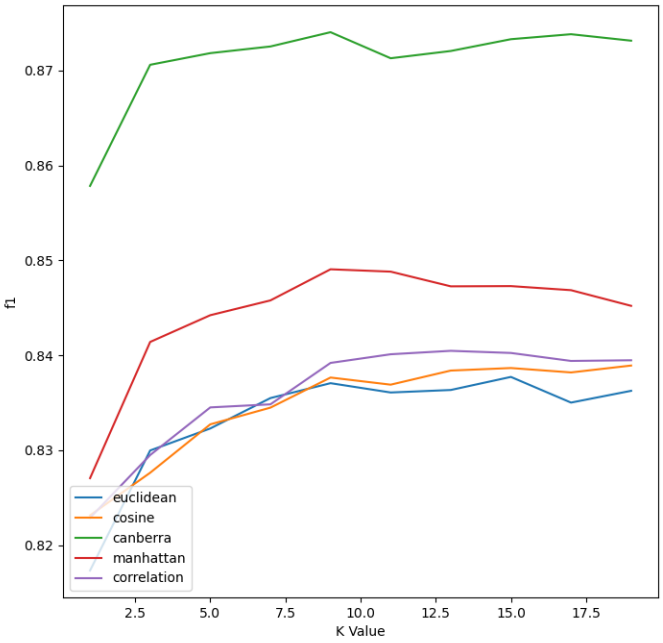

Figure 1: Analyse des métriques avec KNN

1.2.2 Soft voting

Avec les données originales et les algorithmes suivantes:

```
adaboost = AdaBoostClassifier(n_estimators=150)
knn = KNeighborsClassifier(n_neighbors=1)
lgbm = LGBMClassifier(max_depth=18, num_leaves=143,
feature_fraction=0.5398760642626285, bagging_fraction=0.9304436544614162,
learning_rate=0.06525287721325376, max_bin=24, min_data_in_leaf=20,
subsample=0.175744924178873)
rf = RandomForestClassifier(n_estimators=350,
max_depth=45, max_features='auto', bootstrap=True,
min_samples_leaf=1,  min_samples_split=2)
xgboost = XGBClassifier(max_depth=7,n_estimators=157,
learning_rate= 0.23989473738149508,
gamma= 0.7442032316202452, random_state=42)
```

public score: 0.99757, private score: 0.99559

1.3 CNN

On reshape l'ensemble de données en (62000, 12, 18) (batch size, num channel, num features) et applique Conv1d avec les paramètres suivants:

```
conv = nn.Sequential(nn.Conv1d(in_channels=12, out_channels=24,kernel_size=1,
stride=1, padding=0), nn.ReLU(), nn.Flatten())
```

Ensuite, on applique deux couches complètement connectées avec 256 neurones et une couche de softmax.

Résultats: (1) données originales: public score: 0.69601, private score: 0.63845; (2) données normalisées avec la moyenne et l'écart-type du jeu de données d'entraînement: public score: 0.83663, private score: 0.81313.

1.4 LSTM

On reshape l'ensemble de données en (62000, 12, 18) (batch size, sequence length, num features) et normalise les données avec la moyenne et l'écart-type de l'ensemble du jeu de données d'entraînement et applique LSTM avec les paramètres suivants:

1.4.1 one direction LSTM

```
batch_size = 128, input_size = 18, hidden_size = 200,
layer_size = 3, output_size = 2, epochs=50,
learning rate=0.001
```

public score: 0.97810, private score: 0.97792

1.4.2 BiLSTM

```
batch_size = 128, input_size = 18, hidden_size = 100,
layer_size = 2, output_size = 2, epochs=50,
learning rate=0.001
```

public score: 0.96412, private score: 0.97029

2 Résultats

Voici les résultats des approches qu'on a utilisées avec meilleur score public et privé (solution optimale en gras, celle choisi pour le score privé en souligné) :

Table 1: Résultats des Algorithmes sur le Leaderboard

Algorithmes	Public	Private
KNN (neighbours = 1)	∼0.99757	∼0.99669
KNN (ball tree + canberra)	∼0.99273	∼**0.99779**
Hard voting	∼0.98321	∼0.97396
Soft voting	∼0.99757	∼0.99559
CNN	∼0.83663	∼0.81313
LSTM (one)	∼0.97810	∼0.97792
LSTM (bi)	∼0.96412	∼0.97029

3 Discussion

3.1 Normalisation des données

La normalisation des données n'est pas nécessairement obligatoire pour certaines algorithmes de Machine Learning comme KNN, Random Forest et les méthodes de boosting. Nous avons obtenu les meilleurs résultats avec les données originales et l'usage des algorithmes simples telles que KNN (n_neighbours = 1), ce qui pourrait être obtenu par hasard grâce à la bonne qualité des données originales de l'ensemble du jeu de données d'entraînement et la forte similarité entre le train et le test. Par contre, pour certaines algorithmes de ML comme SVM et la plupart des modèles de DL, l'utilisation des données normalisées est en générale nécessaire. Il faut utiliser la moyenne et l'écart-type du train au lieu de manipuler le train et le test en même temps afin d'éviter les problèmes de "data-snooping".

3.2 Hyperparamètres tuning pour les modèles de ML

Nous avons appris de nouvelles techniques pour l'ajustement des hyperparamètres des modèles de ML dans cette compétition. À part des méthodes classiques comme RandomizedSearchCV et GridSearchCV, nous nous avons servi du package "optuna" pour optimiser les hyperparamètres de façon plus concise et rapide.

3.3 Choix des paramètres pour les modèles de DL

Pour le modèle de CNN, le choix des paramètres tels que out_channels, kernel_size, stride, padding joue un rôle très important dans la prédiction pour cette compétition. Par contre, les résultats obtenus par les modèles de LSTM et Bi-LSTM sont moins affectés par le choix de paramètres dans notre cas.

References

https://www.kaggle.com/cdeotte/mnist-perfect-100-using-knn

https://neptune.ai/blog/lightgbm-parameters-guide

https://www.kaggle.com/somang1418/tuning-hyperparameters-under-10-minutes-lgbm

https://lightgbm.readthedocs.io/en/latest/pythonapi/lightgbm.LGBMClassifier.html

https://www.kaggle.com/andradaolteanu/pytorch-rnns-and-lstms-explained-acc-0-99

https://towardsdatascience.com/hyperparameter-tuning-the-random-\forest-in-python-using-scikit-learn-28d2aa77dd74

https://www.kaggle.com/prashant111/a-guide-on-xgboost-hyperparameters-tuning

https://www.kaggle.com/isaienkov/hyperparameters-tuning-techniques

https://xgboost.readthedocs.io/en/stable/parameter.html

Tianxiang Zhang et al., Band Selection in Sentinel-2 Satellite for Agriculture Applications. https://core.ac.uk/download/pdf/288368052.pdf

Kashyap Raiyani et al., Sentinel-2 Image Scene Classification: A Comparison between Sen2Cor and a Machine Learning Approach. https://mdpi-res.com/d_attachment/remotesensing/remotesensing-13-00300/article_deploy/remotesensing-13-00300-v2.pdf

Zhiwei Yi et al., Crop Classification Using Multi-Temporal Sentinel-2 Data in the Shiyang River Basin of China. https://www.mdpi.com/2072-4292/12/24/4052

Claudia Paris et al., Monitoring of agricultural areas by using Sentinel 2 image time series and deep learning techniques. https://www.researchgate.net/publication/346822092_Monitoring_of_agricultural_areas_by_using_Sentinel_2_image_time_series_and_deep_learning_techniques

Charlotte Pelletier et al., Deep Learning for the Classification of Sentinel-2 Image Time Series. https://ieeexplore.ieee.org/document/8900123

Manuel Campos-Taberner et al., Understanding deep learning in land use classification based on Sentinel-2 time series. https://www.nature.com/articles/s41598-020-74215-5