IFT3395 - Compétition

Yutao Zhang 20179488
Seyed Ahmad Farsad 20031258
Bianca Bica 20161056

2021/11/03

Titre du projet

Classifieur d'événements météorologiques extrêmes à partir de données atmosphériques.

Nom de l'équipe sur Kaggle

ML-ABY :

Yutao Zhang 20179488

Seyed Ahmad Farsad 20031258

Bianca Bica 20161056

Introduction

Le problème constitue à classer un ensemble de variables climatiques correspondant à un moment et un emplacement selon i) des conditions standards ii) un cyclone tropical ou iii) une rivière atmosphérique.

Notre approche est de concevoir un classifieur de régression logistique multi-classe basé sur la formule suivante:

$$\hat{\mathbf{y}} = W^T\mathbf{x} + \mathbf{b}$$

En outre, pour augmenter la performance et l'exactitude des prédictions, on entraîne 3 modèles à la fois avec différents ensembles d'entraînement. Par la suite, ils seront assemblés afin d'obtenir une prédiction finale.

Le résultat sur l'ensemble de test fourni (meilleur score sur Kaggle) est de 0.76038.

Feature Design

Sélection de features :

On enlève la date et l'on conserve toutes les autres caractéristiques, en pensant que la date est difficile à régulariser et improbable d'avoir un grand effet sur les événements météorologiques extrêmes.

Pré-traitement :

Pour garder la distribution inhérente des variables, on opte plutôt pour la méthode min-max scaler au lieu de la normalisation gaussienne. La formule utilisée est donc $\bar{f}_i = (f_i - f_{min})/(f_{max} - f_{min})$ pour chaque colonne des différentes caractéristiques.

De plus, pour simplifier le calul, on ajoute une colonne de 1 à la fin pour représenter le biais.

Algorithmes

Étant donné qu'on ait déjà ajouté une colonne de biais, la prédicition est alors calculée par $\mathbf{z} = W^T\mathbf{x} = \mathbf{x}W$.

Sans perte de généralité, notre algorithme calcule en fait $\mathbf{z} = -\mathbf{x}W$ ($Z = -XW$ sous forme de matrice).

La similarité est calculée par : $\mathbb{P}(Y|X,W) = softmax(Z_{X_i,k=Y_i}) = \frac{\exp(Z_{X_i,k=Y_i})}{\sum_{k=0}^{C}\exp(Z_{X_i,k})}$

La fonction de coût est donnée par : $L(W) = -\frac{1}{N}\log\mathbb{P}(Y|X,W) = \frac{1}{N}(Tr(XWY^T) + \sum_{i=1}^{N}\log\sum_{k=0}^{C}\exp((-XW)_{ik})$

En ajoutant la régularisation, on obtient : $f(W) = \frac{1}{N}(Tr(XWY^T) + \sum_{i=1}^{N} \log \sum_{k=0}^{C} \exp((-XW)_{ik}) + \mu||W||^2$ avec le gradient exprimé comme suit : $\frac{1}{N}(X^T(Y - P)) + 2\mu W$.

Méthodologie

Mini-Batch :

On fait la descente de gradient par mini-batch stochastique de taille 32. Pour chaque cycle (*epoch en anglais*) d'apprentissage, on tire aléatoirement un dixième de l'ensemble d'entraînement, qui représentera l'ensemble de validation, puis on fait $\lceil \frac{N}{32} \rceil$ fois descentes de gradient par mini-batch sur le reste de l'ensemble.

Arrêt tôt :

On entraîne au total $max\_iter = 5000$ cycles et à la fin de chacun, on valide le modèle sur l'ensemble de validation : ceci retourne un taux de correction. De cycle en cycle, lorsqu'on obtient un taux plus élevé, on enregistre ce dernier ainsi que le modèle correspondant. Si après $early\_stop$ cycles successifs le taux n'est pas mis à jour, on arrête l'entraînement et on applique le meilleur modèle enregistré jusqu'à présent.

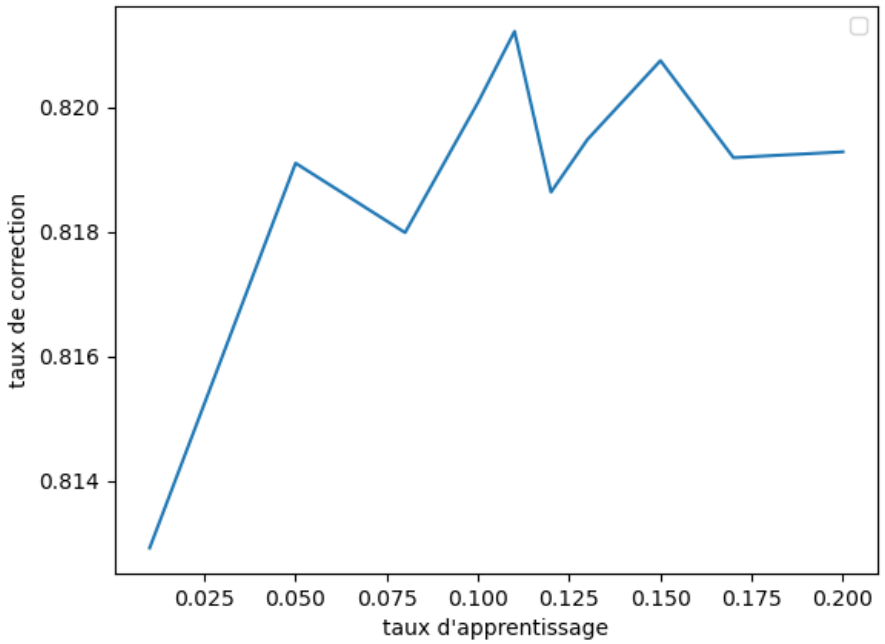
Assemblage :

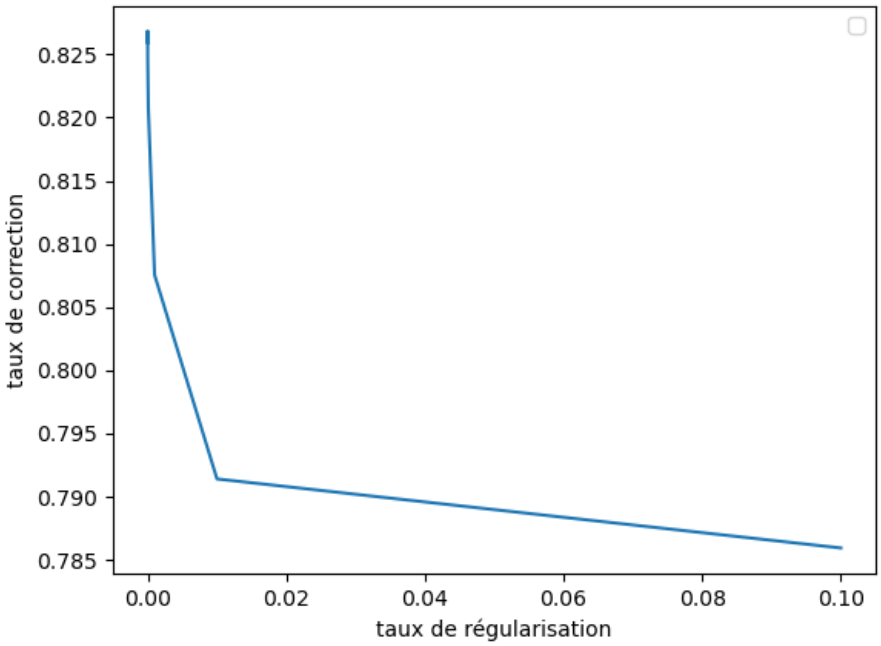
Pour augmenter la performance, on entraîne 3 modèles indépendants à la fois sur différents ensembles d'entraînement. Ensuite, pour déterminer la classe correspondante, on procède par principe de majorité (la classe qui aura la plus grande fréquence sera sélectionnée).

Résultats

Classifieur régression logistique :

Les hyper-paramètres les plus importants sont le taux d'apprentissage (lr) et le taux de régularisation (reg). Le type de régularisation utilisé dans notre modèle est L2, afin de bien gérer la corrélation entre nos variables.





Après plusieurs essais, on obtient le plus haut taux de précision 0.8268 en prenant $lr = 0.11$ et $reg = 0.000001$.

Classifieur naïf de Bayes :

On implémente le classifieur naïf de Bayes avec la fonctionnalité GaussianNB de la librairie sklearn.

Avec la même sélection de features et le prétraitement de la régression logistique, le taux de précision moyen est seulement de 0.5729. En utilisant le modèle fourni par sklearn, il n'y a pas d'hyper-paramètres à ajuster pour obtenir un meilleur résultat. En soit, ce modèle ne possède pas d'hyper-paramètres lorsqu'on utilise une distribution gaussienne, ce qui risque de donner une précision de prédictions plus faible.

SVM :

On implémente le classifieur SVM avec le module SVC de sklearn.

En utilisant la même sélection de features et le prétraitement que dans les deux méthodes précédentes, lorsqu'on utilise le noyau linéaire, on obtient un taux de précision moyen de 0.7997. Si on utilise un noyau polynomial (de degré 3), le taux de précision moyen devient 0.8486. Un des hyper-paramètres les plus importants des SVM qu'on a ajusté est le type de noyau utilisé. Un noyau polynomial retourne une précision plutôt notable comparativement à un noyau linéaire.

Dans la compétion Kaggle, les 3 baselines mises en évidence ont été battues par notre équipe. En observant les trois modèles utilisés, on peut s'apercevoir que le modèle SVM fournit le meilleur résultat (en ajustant l'hyper-paramètre du noyau) et le modèle naïf de Bayes retourne le pire. Le classifieur régression logistique possède un taux de précision moyen assez élevé, surtout après la reconfiguration des taux d'apprentissage et de régularisation. Néanmoins, tous ces modèles pourraient être améliorés pour obtenir un meilleur score, sans toutefois tomber dans des cas de super-apprentissage.

Discussion

Avantages :

(1) Facile et performant : Au lieu d'appliquer One vs. One ou One vs. All, notre algorithme peut réaliser la multi-classification à l'aide d'un seule modèle. Selon nos observations, sa performance est supérieure à celle de l'entraînement de plusieurs classifieurs binaires. De plus, le fait d'essayer trois modèles différents nous indique rapidement quel classifieur est le meilleur pour notre problème.

Inconvénients :

(1) Précision pas idéale : La meilleure précision de notre classifieur est d'environ 0.76. Il y a beaucoup de progrès qui peut être effecuté quant au modèle, notamment dans le choix de celui-ci, les hyperparamètres, la distribution, etc.

Améliorations :

(1) Implémenter un noyau polynomial ou RBF : D'après le résultat montré par le modèle SVM, on voit qu'un noyau polynomial augmente considérablement la performance, impliquant alors que le problème de classification n'est pas linéaire. Pour ce type de problème, le noyau RBF est également une solution à considérer, tout en choisissant la meilleure valeur de gamma possible.

(2) Essayer la régression L1: Puisqu'il y a peut-être des caractéristiques non relatives, on suppose qu'utiliser la méthode de la régression L1 pourrait augmenter la précision.

(3) Implémenter le classifieur naïf de Bayes avec un feature design différent: Ce type de classifieurs suppose que les features avec lesquels on opère sont fortement indépendants entre eux. Ceci n'est pas toujours le cas dans notre contexte et cela explique alors la raison derrière le faible taux de précision obtenu. Idéalement, on ne tiendrait pas compte des variables dépendantes (le niveau de dépendance entre les variables toléré serait fixé à un certain seuil) lors de l'utilisation de ce classifieur. Cependant, cette alternative ne garantit pas un meilleur résultat car la distribution gaussienne n'est pas nécessairement appropriée pour le problème, mais serait possiblement une autre piste à explorer.

Références

Sophia Yang, Multiclass logistic regression from scratch, 18 avril 2021.

https://towardsdatascience.com/multiclass-logistic-regression-from-scratch-9cc0007da372

Pedregosa et al., Scikit-learn: Machine Learning in Python, Naive Bayes, JMLR 12, 2011.

https://scikit-learn.org/stable/modules/naive_bayes.html

Neelam Tyagi, L2 and L1 Regularization in Machine Learning, 28 février 2021.

https://www.analyticssteps.com/blogs/l2-and-l1-regularization-machine-learning

Saumya Awasthi, Seven Most Popular SVM Kernels, 17 décembre 2020.

https://dataaspirant.com/svm-kernels/