Apprentissage automatique - TP3

Classification linéaire, SVC à noyau, Random Forests

Important: Avant de quitter le cours, vous devez enregistrer votre notebook sous le nom `TP3<Prénom> <Nom>.ipynb' et le soumettre sur la plateforme Moodle du cours.

Classifieurs linéaires binaires

Dans la première partie du TP, vous allez travailler avec certains des algorithmes que nous avons vus en cours, pour la classification binaire linéaire. Pour rappel, la classification binaire signifie que la variable cible peut prendre l'une des deux valeurs possibles. La classification linéaire signifie que les points sont séparés à l'aide d'un hyperplan. Les points au-dessus de la ligne appartiennent à la classe positive et les points en dessous de la ligne appartiennent à la classe négative.

Il existe deux classifieurs linéaires populaires :

1. **Régression logistique (Logistic Regression):** `sklearn.linear_model.LogisticRegression`. Le paramètre de régularisation que nous pouvons spécifier est la pénalité L1 ou L2. Par défaut, L2 est utilisée. De plus, nous pouvons spécifier le paramètre de pénalité C pour le terme d'erreur. Par défaut, la valeur 1 est utilisée.
2. **SVC linéaire (Linear SVC):** `sklearn.svm.LinearSVC`. Identique à la régression logistique. Aussi, une implémentation de SVC linéaire (légèrement différente) est fournie par `sklearn.svm.SVC(kernel='linear')`.

Exercice 1

Dans cet exercice, nous allons entraîner et tester différents classifieurs sur un jeu de données factice. L'objectif est de voir si la régression logistique et le SVC linéaire prédisent de manière similaire la classe des points d'entrée et comment la frontière de décision change avec différents paramètres de régularisation.

1. Commencez par créer un jeu de données factice de 30 points en utilisant la fonction `make_blobs()` du module `sklearn.datasets`, avec deux classes (donc deux centres dans les paramètres d'entrée).

2. Nous voulons tracer les lignes de décision pour deux classifieurs dans la même figure. Pour cette raison, nous créons une figure avec deux sous-graphiques en utilisant `plt.subplots(1,2,figsize=(10,3))`.

3. Ensuite, dans chaque axe, nous allons afficher la ligne de décision créée par un classifieur de régression logistique et un classifieur SVC linéaire avec les valeurs par défaut des paramètres. Ajustez les modèles sur le jeu de données factice (caractéristiques et valeurs cibles).

4. Tracez les lignes de décision pour les 2 modèles avec les points de l'ensemble de données factice en utilisant :

   ```
   mglearn.plots.plot_2d_separator(nom_du_classifieur,
   valeurs_de_caractéristique, eps=0.5, ax=Axes_correspondants, alpha=.7 )
   ```

   et

   ```
   mglearn.discrete_scatter(Valeurs_de_Caractéristique1,
   Valeurs_de_Caractéristique2, Valeurs_Cibles, ax=Axes_correspondants)
   ```
   .

5. Affichez également les scores de classification pour les deux classifieurs.

6. Qu'observez-vous ?

7. Maintenant, changez la valeur de pénalité C pour le classifieur SVC en utilisant l'implémentation svm avec un noyau linéaire. Essayez pour C dans l'ensemble {0.01, 1.0, 100.0}. Tracez les trois lignes de décision dans une seule figure avec 3 sous-graphiques. Comment la ligne bouge-t-elle lorsque C augmente ? Quel est l'impact sur le score d'entraînement ?

Remarque pour tracer vous-même la ligne de décision : trouvez les coefficients et l'ordonnée à l'origine de la ligne en utilisant les attributs `coef_` et `intercept_` dans la classe `svm`. La ligne de décision sera de la forme `(ax1 + bx2 + c = 0)`. Utilisez `matplotlib.pyplot.plot(x,f(x),linestyle='solid')` pour tracer la ligne.

SVC à noyau

Jusqu'à présent, nous avons vu des classifieurs linéaires (et quelques cas de classification à marge souple). Scikit-learn implémente également des algorithmes de classification non linéaire, notamment dans la classe des SVM (sklearn.svm.SVC()), en utilisant le concept de noyau. Les noyaux fournis sont 'linéaire', 'poly' (le degré est spécifié dans le paramètre 'degré') et 'rbf'.

Exercice 2

Nous allons créer un nouveau jeu de données factice, de sorte qu'il ne soit pas séparable linéairement. Comme précédemment, utilisez la fonction `make_blobs` mais maintenant avec le paramètre `centers=4`. Rendez le problème binaire, en ne conservant que 2 classes (sur 4). Cela peut être fait en attribuant à la valeur cible la valeur modulo

2 (c'est-à-dire `y=y%2`). Tracez le nuage de points et observez que les points ne sont pas linéairement séparables.

Entraînez un SVC linéaire, un polynomiale et un gaussien (rbf) sur l'ensemble de données.

Comparez les scores d'entraînement pour les trois cas et expliquez pourquoi vous pensez que c'est le résultat.

Classifieurs linéaires multiclasse

La régression logistique et le SVC incluent tous deux une option pour la classification multiclasse. Si l'ensemble de données cible contient plus de deux valeurs, alors la classification multiclasse est activée, et la méthode par défaut utilisée est OneVsRest.

Exercice 3

Maintenant, créez un ensemble de données factice à 3 classes en utilisant `sklearn.datasets.make_blobs` avec un état aléatoire constant (par exemple, `random_state=42`). Réentraînez le modèle sur les nouvelles données.

Combien de lignes de décision attendez-vous avoir été dérivées ? Trouvez les caractéristiques des lignes en utilisant l'attribut `coef_` et `intercept_`.

Utilisez un tracé de dispersion discret `mglearn` pour afficher les points de l'ensemble de données et tracez sur le même diagramme les trois lignes de décision (même procédure qu'auparavant). Expliquez ce que signifient les lignes et comment elles définissent l'espace de classification (c'est-à-dire quelles parties de l'espace correspondent à quelles classes).

Arbres de décision

Exercice 4

Les arbres de décision sont une méthode d'apprentissage supervisé largement utilisée pour la classification et la régression. Dans cet exercice, nous allons explorer les arbres de décision en utilisant scikit-learn.

1. Commencez par importer la classe DecisionTreeClassifier de scikit-learn. Chargez un jeu de données pour la classification et divisez les données en ensembles d'entraînement et de test avec par exemple :

```
from sklearn.datasets import load_iris
```

2. Initialisez un objet DecisionTreeClassifier avec des paramètres de votre choix (par exemple, la profondeur maximale de l'arbre, le critère de division, etc.). Puis ajustez le modèle aux données d'entraînement. Justifiez le choix de vos critères.
3. Évaluez les performances du modèle en utilisant les données de test (par exemple, en calculant la précision, le rappel, le score F1, etc.). Que pouvez-vous en conclure ?
4. Explorez l'importance des caractéristiques en utilisant l'attribut feature_importances_ de l'objet DecisionTreeClassifier.
5. Visualisez la structure de l'arbre de décision en utilisant la fonction plot_tree de scikit-learn ou en exportant l'arbre.
6. Expérimentez avec différents paramètres du DecisionTreeClassifier (par exemple, la profondeur maximale, le critère de division, etc.) et observez comment ils affectent les performances du modèle et l'importance des caractéristiques.

Forêts aléatoires

Exercice 5

Les forêts aléatoires sont une méthode d'apprentissage ensembliste qui combine plusieurs arbres de décision pour effectuer la classification ou la régression. Chaque arbre de décision est formé sur un sous-ensemble aléatoire des données d'entraînement et utilise une fraction aléatoire des caractéristiques pour diviser les nœuds. En fin de compte, les prédictions sont agrégées en prenant la moyenne (pour la régression) ou en effectuant un vote majoritaire (pour la classification) parmi les arbres individuels.

Dans cet exercice, nous allons explorer les forêts aléatoires en utilisant scikit-learn.

1. Commencez par importer la classe RandomForestClassifier de scikit-learn, puis chargez un jeu de données approprié pour la classification. Divisez les données en ensembles d'entraînement et de test.
2. Initialisez un objet RandomForestClassifier avec des paramètres de votre choix (par exemple, le nombre d'arbres dans la forêt, la profondeur maximale des arbres, etc.). Puis ajustez le modèle aux données d'entraînement. Justifiez vos choix de critères.
3. Évaluez les performances du modèle en utilisant les données de test (par exemple, en calculant la précision, le rappel, le score F1, etc.).

4. Explorez l'importance des caractéristiques en utilisant l'attribut feature_importances_ de l'objet RandomForestClassifier.
5. Visualisez la frontière de décision de la forêt aléatoire en 2D (si les données ont moins de 3 caractéristiques) pour mieux comprendre son comportement.
6. Expérimentez avec différents paramètres du RandomForestClassifier (par exemple, le nombre d'arbres, la profondeur maximale, etc.) et observez comment ils affectent les performances du modèle et l'importance des caractéristiques.