HES-SO
MSE

Enseignant: Stefano Carrino
Assistant: Jacky Casas

Multimodal Processing, Recognition and Interaction

Série du 5 novembre 2019
Anomaly detection – SVM

Code et explications

Le code source ainsi que le dataset utilisés dans ce TP sont disponibles sur Moodle dans un fichier zip. La façon d'installer les dépendances requises et de lancer le projet est expliquée dans le fichier README.md.

Tout est également disponible sur Github à l'adresse :

`https://github.com/jacky-humantech/mpri-2019-anomaly-detection`

Exercice 1 OneClassSVM avec Scikit-learn

Dans ce TP, nous utiliserons le classifieur mono-classe `OneClassSVM` de *scikit-learn*[1]. La documentation est accessible à l'adresse suivante :

`http://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html`

Vous pouvez faire quelques tests pour appréhender le fonctionnement des SVM via l'interface dynamique accessible sur le lien suivant :

`https://cs.stanford.edu/~karpathy/svmjs/demo`

Dans ce premier exercice, nous allons prendre le code d'exemple[2] du `OneClassSVM` de la documentation de *scikit-learn*. Ce code n'utilise pas un dataset mais produit lui-même des données afin de les classifier. En l'état, le code produit un graphique qui montre ce qui se produit, comme on le voit à la figure 1.

Dans cet exercice d'introduction, répondez aux questions suivantes dans le but d'appréhender le fonctionnement de la SVM :

1. `http://scikit-learn.org`
2. `http://scikit-learn.org/stable/auto_examples/svm/plot_oneclass.html`

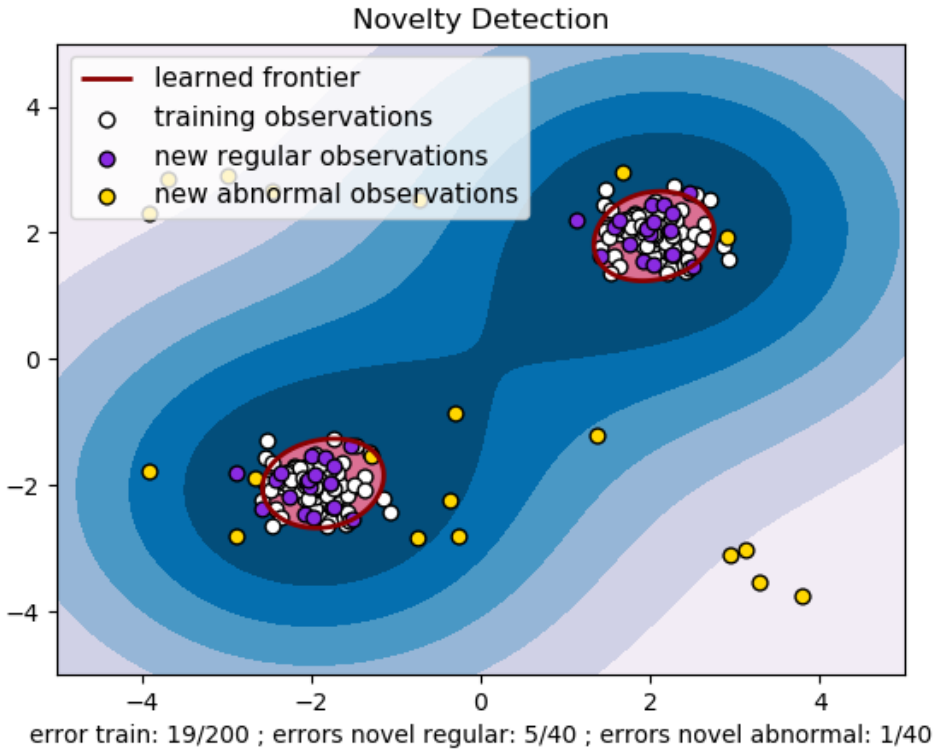

FIGURE 1 – Graphique produit par le code d'exemple. Source : `http://scikit-learn.org/stable/_images/sphx_glr_plot_oneclass_001.png`

a) Comment les échantillons sont générés (astuce, manière de faire, etc.) ? Pourquoi y a-t-il deux pôles de concentration de points (type de distribution, etc.) ?

b) Dans le code ci-dessous, à quoi correspondent les variables `X_train`, `X_test` et `X_outliers`. Pourquoi n'a-t-on pas uniquement le *training set* et le *test set* ?

```
clf = svm.OneClassSVM(nu=0.1, kernel="rbf", gamma=0.1)
clf.fit(X_train)
y_pred_train = clf.predict(X_train)
y_pred_test = clf.predict(X_test)
y_pred_outliers = clf.predict(X_outliers)
```

c) Pourquoi ne donne-t-on pas les étiquettes (ou labels) à la fonction `fit()` d'apprentissage ?

d) Tester d'autres noyaux (kernel) pour le classifieur et garder celui qui donne les meilleurs scores. Modifier ensuite les différents paramètres afin d'améliorer les prédictions (*gamma*, *nu*, et d'autres en fonction du noyau choisi - voir documentation).

Exercice 2 Novelty detection

Dans l'exercice 2, nous allons utiliser un vrai set de données afin de rendre la tâche plus authentique. Le dataset choisi est *Breast Cancer Wisconsin*[3]. Il contient des enregistrements physiologiques relatifs à des cancers du sein, certains sont bénins et d'autres malins. Nous considérons les enregistrements qui concernent des cancers malins comme les outliers. Ils sont présents en plus petite quantité dans le jeu de données.

La structure du classifieur est fournie. Certaines parties sont manquantes et c'est ce que nous devons compléter ici.

Tâche 1 :

La première étape consiste à charger le dataset et à nettoyer les données manquantes. Cette étape est déjà faite. Dans la deuxième étape, le but est de comprendre les données que nous avons à disposition. Pour cela, rien de mieux que de les visualiser grâce à toutes sortes de graphiques. Trois méthodes sont déjà à votre disposition, les voici :

a) `plot_3d()` : Prend 3 caractéristiques du dataset (à vous de choisir celles que vous voulez) et affiche un graphe en 3D des données (voir figure 2). Les trois axes sont les 3 caractéristiques choisies. Vous pouvez bouger le graphique avec votre souris.

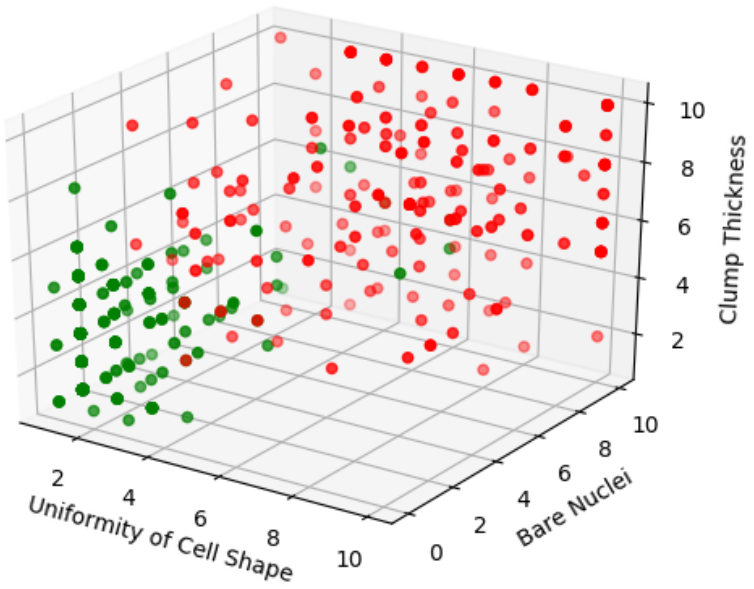

FIGURE 2 – Graphique nuage de points du dataset en fonction de 3 caractéristiques

b) `plot_scatter_matrix()` : La Scatter Matrix est un ensemble de graphiques qui consiste à comparer chaque caractéristique une à une (avec un graphique en nuage de points). Dans la diagonale se trouve un histogramme de la répartition des échantillons dans la caractéristique correspondante.

3. `http://odds.cs.stonybrook.edu/breast-cancer-wisconsin-original-dataset`

c) `plot_parallel_coordinates()` : Ce graphique est une autre façonde représenter un dataset qui contient plus de 3 caractéristiques.

La fonction `plot_3d()` fonctionne déjà, vous pouvez la tester dès à présent.
Par contre `plot_scatter_matrix()` et `plot_parallel_coordinates()` non. Elles utilisent la fonction `extract_features()` qui n'est pas implémentée.

Implémenter la fonction `extract_features()`. Cette méthode prend en argument le dataset en format `DataFrame` (voir lib *pandas*[4]) et retourne également un `DataFrame`. Son fonctionnement est similaire à la fonction `extract_labels()`.

NB : Dans la fonction d'extraction de caractéristiques, vous pouvez toutes les prendre, ou alors choisir uniquement celles que vous voulez. Il peut être utile de ne pas toutes les visualiser afin que les graphiques soient plus lisibles.

Tâche 2 :

La prochaine étape est le traitement des données. **Séparez le dataset en deux parties selon la classe.** En effet, nous voulons faire de la *"novelty detection"*, et donc le set d'entraînement ne doit pas contenir d'exception.

Ensuite nous séparons le set contenant seulement les données de cancer bénin en deux : *training set* (2/3) et *test set* (1/3).

NB : Dans ces sets de données, nous voulons extraire les caractéristiques utiles. Nous utilisons donc encore la méthode ***`extract_features()`*** *de la tâche 1. Pensez donc à y mettre les bonnes caractéristiques.*

Tâche 3 :

Le classifieur utilisé est `OneClassSVM`. Il n'est pas optimisé pour le moment. Nous pouvons prédire les résultats sur le *training set* (le set qui a servi à l'entraînement) pour voir si l'entraînement s'est bien passé. Mais nous voulons surtout tester le *test set* (contenant uniquement des échantillons bénins) ainsi que le set contenant les *outliers* (échantillons malins). Pour évaluer les résultats, les fonctions `print_errors_in_prediction()` et `print_scores()` sont à votre disposition.

Les paramètres utilisés pour le modèle sont ceux par défaut (`gamma`, `nu`, ...). Votre but est de **régler les paramètres du classifieur** afin d'optimiser le nombre d'échantillons bénins détectés comme tel, tout en optimisant le nombre d'échantillons malins détectés comme tel (**optimisation du F1-score**).

NB : Pensez à adapter les caractéristiques que vous prenez en compte. Il est possible que certaines d'entre elles n'aident pas la modèle.

4. `http://pandas.pydata.org/pandas-docs/stable`

Exercice 3 Outlier detection

Jusqu'à présent, nous avons entraîné le modèle uniquement avec des échantillons bénins. Dans un cas pratique réel, il se peut que des échantillons malins se glissent dans le set d'entraînement. C'est ce que nous allons simuler dans cet exercice.

Reprenez votre code de l'exercice 2 avec les meilleurs réglages. Votre but est d'ajouter de plus en plus d'outliers dans le set d'entraînement et de voir l'évolution du score de classification.

Faites un graphique qui montre cette évolution (f1-score en fonction de la proportion d'*outliers* dans le set d'entraînement).

Expliquez et discutez vos résultats.

Que faut-il rendre ? Dans une archive zip :

— le rapport au format PDF contenant les réponses aux questions et vos analyses
— le code Python final de l'exercice 2 et un autre pour l'exercice 3