Gradient boosting

Data science
Master 2 ISiDIS
2015 / 2016

Prise en main et exemples d'utilisation

1. Lancer ipython et le notebook :
   `ipython notebook --pylab=inline`

2. Créer un nouveau notebook dans lequel vous pouvez exécuter des commandes python. Dans le pylab, les bibliothèques numpy, scipy, matplotlib, pandas, sklearn sont automatiquement importée.
3. Suivre l'exemple de classification avec le gradient boosting de la page 20 des slides de Peter Prettenhofer et Gilles Louppe disponibles à l'url `http://orbi.ulg.ac.be/bitstream/2268/163521/1/slides.pdf`.
4. Suivre l'exemple donné par la librairie scikit-learn sur la régularisation du gradient boosting :
   `http://scikit-learn.org/stable/auto_examples/ensemble/plot_gradient_boosting_regularizati`
   `html#example-ensemble-plot-gradient-boosting-regularization-py`

Exemple de régression et paramètres

1. Quels sont les principaux paramètres réglables du gradient boosting de la bibliothèque scikit-learn ?
2. Importer le jeu de données 'boston' et créer des jeux de test d'apprentissage et de test à l'aide des lignes suivantes. Que détermine le nombre `0.9` ?

```
from sklearn import datasets
from sklearn.utils import shuffle
X, y = shuffle(boston.data, boston.target, random_state=13)
X = X.astype(np.float32)
offset = int(X.shape[0] * 0.9)
X_train, y_train = X[:offset], y[:offset]
X_test, y_test = X[offset:], y[offset:]
```

3. Calculer la régression gradient boosting avec les paramètres suivants :

```
params = {'n_estimators': 500, 'max_depth': 4, 'min_samples_split': 1,
'learning_rate': 0.01, 'loss': 'ls'}
```

4. Tracer les courbes d'erreur en apprentissage et en test au cours des itérations de l'algorithme à l'aide de la fonction suivante :

```
def diagnose(X, gb, params):
    test_score = np.zeros((params['n_estimators'],), dtype = np.float64)
    for i, y_pred in enumerate(gb.staged_decision_function(X_test)):
        test_score[i] = gb.loss_(y_test, y_pred)
    pl.figure(figsize=(12, 9))
    pl.title('Deviance')
    pl.plot(np.arange(params['n_estimators']) + 1, gb.train_score_,
     'b-', label = 'Training Set Deviance')
    pl.plot(np.arange(params['n_estimators']) + 1, test_score,
     'r-', label = 'Test Set Deviance')
    pl.legend(loc='upper right')
    pl.xlabel('Boosting iterations')
    pl.ylabel('Deviance')
    pl.show()
```

5. Modifier les paramètres du gradient boosting et observer les effets sur les courbes d'apprentissage et de test.
6. Déterminer les (hyper)paramètres optimaux du gradient boosting à l'aide de la technique exposé page 27 des slides de Peter Prettenhofer et Gilles Louppe.