TD classification linéaire

via package scikit-learn (=module sklearn)

1 Classification pour identification de langue

Détaillez comment construire un identifieur de langue parmi C langues possibles, de type classifieur supervisé appris via perceptron, en représentant un texte dans l'espace vectoriel des ngrammes de caractères (typiquement n=2).

Quelle différence faites-vous avec un identifieur de langue construit comme suit (cf. projet L3 pour certains) :

- on dispose d'un corpus $corp_i$ pour la langue i, pour i de 1 à C
- soit un texte P dont on veut identifier la langue
- soit ϕ la fonction associant un texte ou corpus o à sa représentation vectorielle de P : nb d'occurrences des bigrammes de caractères contenus ds o
- on prédit pour P la langue $\hat{\imath} = \text{argmax}_i \cos(\phi(P), \phi(corp_i))$

2 Régression linéaire via sklearn

Il s'agit d'utiliser le module python scikit-learn qui implémente divers d'algorithmes d'apprentissage (à utiliser pour modèles linéaires et log-linéaires, moins adapté pour les modèles avec apprentissage profond)

Les parties classification et régression de la page d'accueil pointent vers la même page concernant l'apprentissage supervisé :
https://scikit-learn.org/stable/supervised_learning.html#supervised-learning

Il y a beaucoup de tutoriels en ligne, en particulier :

- http://scikit-learn.org/stable/tutorial/basic/tutorial.html
- https://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html

La méthode qui nous intéresse est « Ordinary least squares » = identification des paramètres d'une régression linéaire via la méthode des moindres carrés.
On propose ici d'aller directement à l'exemple suivant de régression linéaire (sur un pb non TAL) :
https://scikit-learn.org/stable/auto_examples/linear_model/plot_ols.html#sphx-glr-auto-examples-linear-model-plot-ols-py
qui utilise la classe LinearRegression :
https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

Importez le **notebook** disponible en bas de la page et jouez avec pour répondre aux questions suivantes :

- quel est le type de diabetes_X ?
- quel est le type et la shape de diabètes_X.data
- Où sont les valeurs gold ?
- Quelles sont les méthodes pour l'apprentissage et pour la prédiction et que prennent-elles en entrée ?
  - ces méthodes sont toujours les mêmes pour tous les modèles sklearn !
- Que fait la ligne « diabetes_X = diabetes.data[:, np.newaxis, 2] » ?
- Quelle est la métrique d'évaluation ?
- En plus : cherchez dans la doc l'explication de r2_score

3 Modules de chargement de textes

Voir le notebook spécifique sur les modules sklearn de chargement de textes
cf.
https://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_extraction.text

4 Classification linéaire via sklearn

On reprend de nouveau le jeu de données reuters (small / medium / reuters pour les tests de différentes tailles), pour tester différents classifieurs, mais on part maintenant d'un format « onedocperline »

ATTENTION : un doc peut être associé à plusieurs classes (séparées par virgule)
=> vous dupliquerez artificiellement ces docs autant de fois que nécessaire pour se ramener au cas mono-label.

Ecrivez un programme qui :

- charge ces documents dans matrices de train / de test,
- apprend sur train un perceptron, et le teste sur test
  https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Perceptron.html
- étudiez sklearn.metrics.accuracy_score pour faire afficher la précision obtenue sur le test, sur le train

En plus :

- étudiez la page
  http://scikit-learn.org/stable/supervised_learning.html#supervised-learning
  Quels sont les algorithmes proposés dont ns avons parlé en cours ?
- testez d'autres algos que perceptron (SVM...)
  - (tous les classifieurs proposés ont les méthodes `fit` et `predict` ...)
- implémentez une recherche en grille pour les hyperparamètres du SVM (pour cela étudiez `sklearn.model_selection.GridSearchCV`)
  - `import sklearn.model_selection`
  - `help(sklearn.model_selection.GridSearchCV)`
- cherchez comment faire de la validation croisée … etc...