Notas de clase para la implementación

Aprendizaje Automático
2016

Menu

- Cross validation
- GridSearch
- Pipeline
- Feature Extraction

Cross validation

Sklearn implementa varias formas en el módulo cross_validation

Cross validation

Sklearn implementa varias formas en el módulo cross_validation

- K-folds

Link: http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.KFold.html#sklearn.cross_validation.KFold

Idea: Divide la muestra de n instancias en K folds

class sklearn.cross_validation.**KFold**(*n*, *n_folds=3*, *shuffle=False*, *random_state=None*)

```
>>> from sklearn.cross_validation import KFold
>>> X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
>>> y = np.array([1, 2, 3, 4])
>>> kf = KFold(4, n_folds=2)
>>> for train_index, test_index in kf:
...   print("TRAIN:", train_index, "TEST:", test_index)
...   X_train, X_test = X[train_index], X[test_index]
...   y_train, y_test = y[train_index], y[test_index]
TRAIN: [2 3] TEST: [0 1]
TRAIN: [0 1] TEST: [2 3]
```

Cross validation

Sklearn implementa varias formas en el módulo cross_validation

- Stratified KFold

Link: http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.StratifiedKFold.html

Idea: Divide la muestra en K folds intentando respetar la distribución de las **CLASES** para cada fold

class sklearn.cross_validation.**StratifiedKFold**(*y*, *n_folds=3*, *shuffle=False*, *random_state=None*)

```
>>> from sklearn.cross_validation import StratifiedKFold
>>> X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])
>>> y = np.array([0, 0, 1, 1])
>>> skf = StratifiedKFold(y, n_folds=2)
>>> for train_index, test_index in skf:
...    print("TRAIN:", train_index, "TEST:", test_index)
...    X_train, X_test = X[train_index], X[test_index]
...    y_train, y_test = y[train_index], y[test_index]
TRAIN: [1 3] TEST: [0 2]
TRAIN: [0 2] TEST: [1 3]
```

Cross validation

Sklearn implementa varias formas en el módulo cross_validation

- Más sabores en http://scikit-learn.org/stable/modules/cross_validation.html

- **Atajo:** cross_val_score

sklearn.cross_validation.**cross_val_score**(*estimator*, *X*, *y=None*, *scoring=None*, *cv=None*, *n_jobs=1*,*verbose=0*, *fit_params=None*, *pre_dispatch='2*n_jobs'*)

```
1.  import sklearn.tree
2.  import sklearn.cross_validation
3.  clf= sklearn.tree.DecisionTreeClassifier()
4.  scores = cross_validation.cross_val_score(clf, X, y, cv=5)
5.  scores
6.  array([ 0.96...,  1.  ...,  0.96...,  0.96...,  1.        ])
```

Grid Search

¿Cómo elegimos el mejor clasificador con el mejor hiper parámetro?

Grid Search

¿Cómo elegimos el mejor clasificador con el mejor hiper parámetro?

```
from sklearn.grid_search import GridSearchCV
from sklearn.tree import DecisionTreeClassifier

clf=DecisionTreeClassifier()
 param_grid = {"max_depth": [3, None],
               "max_features": [1, 3, 10],
               "min_samples_split": [1, 3, 10],
               "criterion": ["gini", "entropy"]}
grid_search = GridSearchCV(clf, param_grid=param_grid)
grid_search.fit(X, y)

grid_search.best_score_
0.92821368948247074

grid_search.best_params_
{'criterion': 'entropy',
 'max_depth': None,
 'max_features': 10,
 'min_samples_split': 1}
```

http://scikit-learn.org/stable/modules/generated/sklearn.grid_search.GridSearchCV.html#sklearn.grid_search.GridSearchCV

Pipeline

¿Cómo automatizamos: transformar información -> clasificar?

Pipeline

¿Cómo automatizamos: transformar información -> clasificar?

Supongamos que sistematicamente queremos

- Hacer transformaciones a la matriz de instancias
- Clasificar

```
1.  from sklearn.tree import DecisionTreeClassifier
2.  from sklearn.pipeline import Pipeline
3.  from sklearn.decomposition import PCA
4.  from sklearn.preprocessing import Binarizer
5.  from sklearn.cross_validation import cross_val_score
6.  # Creo un proceso de pipeline
7.  estimators = [('reduce_dim', PCA()),('binarizer', Binarizer(copy=True, threshold=0.5)), ('dtC', DecisionTreeClassifier())]
8.  clf = Pipeline(estimators)
9.  scores = cross_val_score(clf,X,y,cv=10)
10. Scores
11. array([ 0.70810811,  0.63387978,  0.68508287,  0.73333333,  0.69832402,
            0.74301676,  0.73743017,  0.70786517,  0.73446328,  0.71022727])
```

Pipeline

¿Cómo automatizamos: transformar información -> clasificar?

Links:

http://scikit-learn.org/stable/modules/pipeline.html

http://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html

Text Feature Extraction

Hay infinitos módulos de Python que implementan extracción de features para texto, algunos:

Sklearn: http://scikit-learn.org/stable/modules/feature_extraction.html

Gensim: https://radimrehurek.com/gensim/

Natural Language Toolkit: http://www.nltk.org/

Etc…