Supplementary material for functions in Sklearn

February 11, 2018

This supplementary material briefly illustrates the functions used in the previous labs. For more details, please see “http://scikit-learn.org”. Note that in this material, we assume every student knows Python’s class mechanism.

- Linear models (“from sklearn import linear_model as lm”)
  - lm.LinearRegression(fit_intercept=True, normalize=False, copy_X=True, n_jobs=1)
    - Common parameters
      - fit_intercept: whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations.
      - normalize: this parameter is ignored when fit_intercept is set to False. If True, the regressors X will be normalized before regression by subtracting the mean and dividing by the L2-norm.
    - Common attributes
      - coef_: parameter vector.
      - intercept_: intercept
    - Common methods
      - fit(X, y): fit linear model.
      - predict(X): predict using the linear model.
  - lm.LogisticRegression(penalty=’l2’, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver=’liblinear’, max_iter=100, multi_class=’ovr’, verbose=0, warm_start=False, n_jobs=1)
    - Common parameters
      - C: inverse of regularization strength; must be a positive float. Smaller values specify stronger regularization.
      - fit_intercept: if a constant (bias or intercept) should be added to the decision function.

        - multi_class: multi-class option can be either ‘ovr’ or ‘multinomial’. If the option chosen is ‘ovr’, then a binary problem is fit for each label. Else the loss minimised is the multinomial loss fit across the entire probability distribution. Does not work for liblinear solver.
    - Common attributes
        - coef_: parameter vector.
        - intercept_: intercept
    - Common methods
        - fit(X, y): fit the model according to the given training data.
        - predict(X): predict class labels for samples in X.
        - predict_proba(X): probability estimates.
- lm.Lars(fit_intercept=True, verbose=False, normalize=True, precompute=’auto’, n_nonzero_coefs=500, eps=2.2204460492503131e-16, copy_X=True, fit_path=True, positive=False)
    - Common parameters
        - fit_intercept : if a constant (bias or intercept) should be added to the decision function.
        - normalize: this parameter is ignored when fit_intercept is set to False. If True, the regressors X will be normalized before regression by subtracting the mean and dividing by the L2-norm.
        - fit_path: if True the full path is stored in the coef_path_ attribute.
        - n_nonzero_coefs : target number of non-zero coefficients.
    - Common attributes
        - active_: indices of active variables at the end of the path.
        - coef_path_: the varying values of the coefficients along the path.
        - coef_: parameter vector.
        - intercept_: intercept
    - Common methods
        - fit(X, y): fit linear model.
        - predict(X): predict using the linear model.

- Imputation (“from sklearn.preprocessing import Imputer”)
  - Imputer(missing_values='NaN', strategy='mean', axis=0, verbose=0, copy=True)
    - Common parameters
      - missing_values: the placeholder for the missing values. All occurrences of missing_values will be imputed. For missing values encoded as np.nan, use the string value “NaN”.
      - strategy: the imputation strategy.
        1. If “mean”, then replace missing values using the mean along the axis.
        2. If “median”, then replace missing values using the median along the axis.
        3. If “most_frequent”, then replace missing using the most frequent value along the axis.
      - axis: the axis along which to impute.
        1. If axis=0, then impute along columns.
        2. If axis=1, then impute along rows.
      - fit_intercept: if a constant (bias or intercept) should be added to the decision function.
      - multi_class: multi-class option can be either ‘ovr’ or ‘multinomial’. If the option chosen is ‘ovr’, then a binary problem is fit for each label. Else the loss minimised is the multinomial loss fit across the entire probability distribution. Does not work for liblinear solver.
    - Common methods
      - fit(X): fit the imputer on X.
      - fit_transform(X): fit to data, then transform it.

- PCA (“from sklearn.decomposition import PCA”)
  - PCA(n_components=None, copy=True, whiten=False, svd_solver=’auto’, tol=0.0, iterated_power=’auto’, random_state=None)
    - Common parameters
      - n_components: number of components to keep. If n_components is not set all components are kept.
      - fit_intercept: if a constant (bias or intercept) should be added to the decision function.
      - multi_class: multi-class option can be either ‘ovr’ or ‘multinomial’. If the option chosen is ‘ovr’, then a binary problem is fit for each label. Else the loss minimised is the multinomial loss fit across the entire probability distribution. Does not work for liblinear solver.
    - Common attributes
      - components_: principal axes in feature space, representing the directions of maximum variance in the data.
      - explained_variance_: the amount of variance explained by each of the selected components.
    - Common methods
      - fit(X): fit the model with X.
      - fit_transform(X): fit the model with X and apply the dimensionality reduction on X.

- Tree ("from sklearn import tree")
  - tree.DecisionTreeClassifier(criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False)
    - Common parameters
      - criterion : the function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.
      - splitter: the strategy used to choose the split at each node. Supported strategies are "best" to choose the best split and "random" to choose the best random split.
      - max_depth: the maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.
      - min_impurity_decrease: a node will be split if this split induces a decrease of the impurity greater than or equal to this value.
      - min_impurity_split: threshold for early stopping in tree growth.
    - Common attributes
      - feature_importances_: the feature importances. The higher, the more important the feature.
      - n_classes_: the number of classes (for single output problems), or a list containing the number of classes for each output (for multi-output problems).
    - Common methods
      - fit(X, y): build a decision tree classifier from the training set (X, y).
      - predict(X): predict class or regression value for X
      - decision_path(X): return the decision path in the tree
      - predict_proba(X): predict class probabilities of the input samples X.