Linear Regression

Advantages	Disadvantages
Linear Regression is simple to implement and easier to interpret the output coefficients.	On the other hand in linear regression technique outliers can have huge effects on the regression and boundaries are linear in this technique.
When you know the relationship between the independent and dependent variable have a linear relationship, this algorithm is the best to use because of it's less complexity to compared to other algorithms.	Diversely, linear regression assumes a linear relationship between dependent and independent variables. That means it assumes that there is a straight-line relationship between them. It assumes independence between attributes.
Linear Regression is susceptible to over-fitting but it can be avoided using some dimensionality reduction techniques, regularization (L1 and L2) techniques and cross-validation.	But then linear regression also looks at a relationship between the mean of the dependent variables and the independent variables. Just as the mean is not a complete description of a single variable, linear regression is not a complete description of relationships among variables.

From Sklearn.linear_model import LinearRegression

LinearRegression fits a linear model with coefficients w = (w1, …, wp) to minimize the residual sum of squares between the observed targets in the dataset, and the targets predicted by the linear approximation.

Parameters

fit_intercept*bool, default=True*

Whether to calculate the intercept for this model. If set to False, no intercept will be used in calculations (i.e. data is expected to be centered).

normalize*bool, default=False*

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. If you wish to standardize, please use `StandardScaler` before calling `fit` on an estimator with `normalize=False`.

copy_X*bool, default=True*

If True, X will be copied; else, it may be overwritten.

n_jobs*int, default=None*
The number of jobs to use for the computation. This will only provide speedup for n_targets > 1 and sufficient large problems. `None` means 1 unless in a `joblib.parallel_backend` context. `-1` means using all processors. See Glossary for more details.

positive*bool, default=False*
When set to `True`, forces the coefficients to be positive. This option is only supported for dense arrays.

New in version 0.24.

Attributes

coef_*array of shape (n_features,) or (n_targets, n_features)*
Estimated coefficients for the linear regression problem. If multiple targets are passed during the fit (y 2D), this is a 2D array of shape (n_targets, n_features), while if only one target is passed, this is a 1D array of length n_features.

rank_*int*
Rank of matrix `X`. Only available when `X` is dense.

singular_*array of shape (min(X, y),)*
Singular values of `X`. Only available when `X` is dense.

intercept_*float or array of shape (n_targets,)*
Independent term in the linear model. Set to 0.0 if `fit_intercept = False`.

See also

`Ridge`
Ridge regression addresses some of the problems of Ordinary Least Squares by imposing a penalty on the size of the coefficients with l2 regularization.

`Lasso`
The Lasso is a linear model that estimates sparse coefficients with l1 regularization.

`ElasticNet`
Elastic-Net is a linear regression model trained with both l1 and l2 -norm regularization of the coefficients.

Notes

From the implementation point of view, this is just plain Ordinary Least Squares (scipy.linalg.lstsq) or Non Negative Least Squares (scipy.optimize.nnls) wrapped as a predictor object.

Example..

Click Here

Methods we Can Use

`fit`(X, y[, sample_weight])	Fit linear model.
`get_params`([deep])	Get parameters for this estimator.
`predict`(X)	Predict using the linear model.
`score`(X, y[, sample_weight])	Return the coefficient of determination R2 of the prediction.
`set_params`(**params)	Set the parameters of this estimator.

Parameters:

X*{array-like, sparse matrix} of shape (n_samples, n_features)*

Training data

Y:*array-like of shape (n_samples,) or (n_samples, n_targets)*
Target values. Will be cast to X's dtype if necessary

sample_weight:*array-like of shape (n_samples,), default=None*
Individual weights for each sample

Score:

Return the coefficient of determination R^2 of the prediction.

The coefficient R^2 is defined as $(1 - \frac{u}{v})$, where u is the residual sum of squares `((y_true - y_pred) ** 2).sum()` and v is the total sum of squares `((y_true - y_true.mean()) ** 2).sum()`. The best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse). A constant model that always predicts the expected value of `y`, disregarding the input features, would get a R^2 score of 0.0.

Problem we can got :

Over fitting

Under fitting

For that we can see from below link

For Link → [Click Here]

Thank You

[Ranjit Maity]