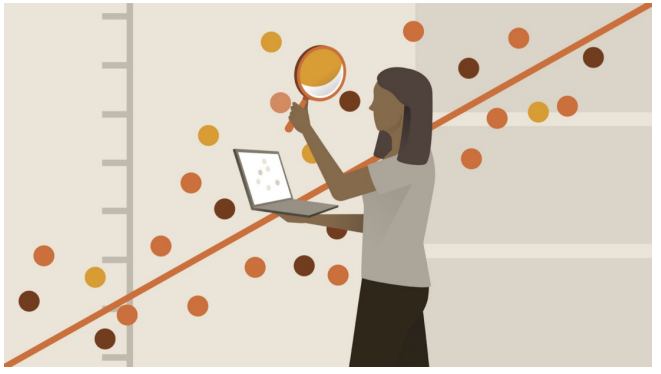
Linear, Polynomial and Logistic Regression



Lluís Talavera

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

ETSEIB

Simple Linear Regression (SLR)

Models the relationship between two variables by fitting a linear equation to the examples.

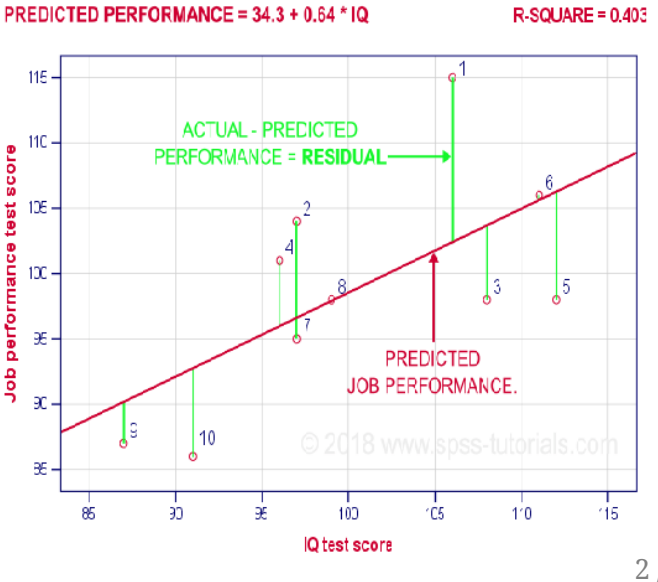
$$Y = \beta_0 + \beta_1 X + \epsilon$$

β_0 is the intercept

β_1 is the slope

residuals: distance between each example and the fitted line.

SLR tries to find the line closests to the set of examples, i.e., the one which **minimizes the sum of squared residuals**.



Multiple Linear Regression (MLR)

We can generalize the previous formula for n variables trying to fit a hyperplane:

$$Y = \beta_0 + \beta_1 X_1 + \dots + \beta_n X_n + \epsilon$$

There are other types of multiple linear regression models:

- Ridge regression: imposes a penalty on the size of coefficients (L2 regularization).
- LASSO regression: can shrink coefficients to zero, so they are effectively discarded (L1 regularization).
- Elastic net regression: A combination of Ridge and LASSO approaches.

There are a lot model assumptions: linearity, homoscedasticity of residuals, normality of residuals, etc. They are required for inference purposes (e.g., confidence intervals) but no always for prediction.

Polynomial Regression

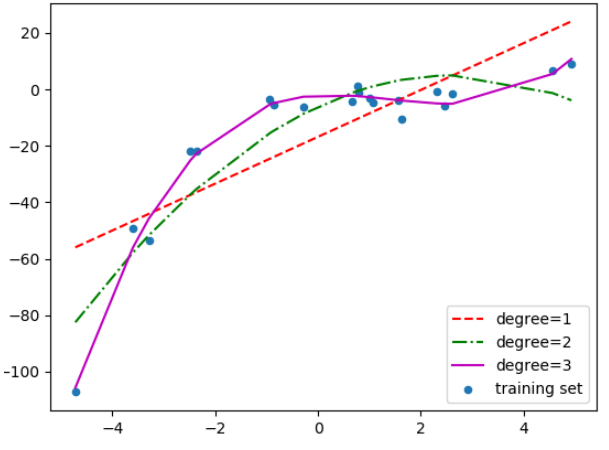
Uses non-linear functions (polynomials) to fit the data by introducing extra predictors obtained by raising the linear predictors to a certain power.

For example, for quadratic regression the formula would be:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^{2}$$

And in general, for a polynomial of degree h:

$$ Y = \beta_0 + \sum_{i=1}^h \beta_i X^i$$



(source)

Polynomial Regression

- As in MLR, it can be generalized for multiple predictors.
- Risk of too much fit, poor generalization (*overfitting**)
- Actually, we are using feature engineering (raising existing features to an exponent) and then fitting a MLR so we could think about other choices besides polynomials:

Square root

$$Y = \beta_0 + \beta_1 \sqrt{X}$$

Logarithm

$$Y = \beta_0 + \beta_1 \log{X}$$

* We will elaborate later about this topic.

Logistic Regression

The name might be confusing because it is a *classification* algorithm.

Predicts a binary output fitting a "S" shaped sigmoid (logistic) function:

$$s(z) = \large{\frac{1}{1+e^{-z}}}$$

The result is the probability of an example belonging to one of two classes. For classification, the default decision threshold is 0.5:

if prob(1)>=0.5 return 1

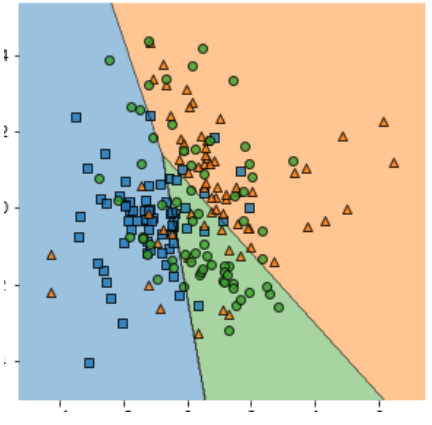
else return 0

Logistic Regression

In the previous equation, z is a linear combination on one or more predictors:

$$z = \beta_0 + \beta_1 X_1 + \dots + \beta_n X_n$$

LR uses *maximum likelihood estimation (MLE)*, i.e., chooses values for the parameters that maximize the probability of observing the class y given the features X. This probability is summarized in the *likelihood function*.



It is a **linear classifier**.

sklearn

```
from sklearn.linear_model import LinearRegression
```

```
from sklearn.preprocessing import PolynomialFeatures

# Generate a new feature matrix consisting of all polynomial combinations
# of the features with degree less than or equal to the specified degree
trans = PolynomialFeatures(degree=2)
X_poly = trans.fit_transform(X)
```

```
from sklearn.linear_model import LogisticRegression
```