12_regression_theory

May 7, 2025

1 Linear Regression - Theory

Ref: "Statistical Methods for the Earth Scientist: An Introduction" R. Till, 1974

This will be a brief, non-statistically thorough, introduction to the terminology and mathematical syntax of linear regression.

1.1 Scalar, Vector and Matrix Nomenclature being used

Variables - lower case: x

A Vector - lower case bold: $\mathbf{x}$

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

$$\mathbf{x}^T = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \end{pmatrix}$$

Matrices - upper case bold: $\mathbf{X}$ of size m rows by n columns

$$\mathbf{X_{m \times n}} = \begin{pmatrix} \mathbf{x_1}^T \\ \mathbf{x_2}^T \\ \vdots \\ \mathbf{x_m}^T \end{pmatrix}$$

1.2 What is Linear Regression?

Linear regression is one of the most fundamental and widely used supervised machine learning algorithms. It models the relationship between a dependent variable (target) and one or more *independent* variables (features) by fitting a linear equation to the observed data.

The basic form of a linear regression equation is:

$$y = \beta_0 + x_1\beta_1 + x_2\beta_2 + ... + x_n\beta_n + \epsilon$$

Where: - y is the dependent variable that we're trying to predict - x_1, x_2, ..., x_n are the independent variables - β_0 is the y-intercept (bias) - β_1, β_2, ..., β_n are the coefficients (weights) for each feature - ϵ is the error term

In matrix form:

$$\mathbf{y} = \mathbf{X}\beta + \epsilon$$

$\mathbf{y}$ A (column) vector of responses $\mathbf{X}$ A matrix of independent variables (one row for each sample) β A vector of coefficients.

Equation to solve for β:

$$\beta = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

1.3 Importance in Machine Learning

Linear regression is important for several reasons:

1. **Simplicity**: It's easy to understand and implement
2. **Interpretability**: The coefficients directly tell us how much each feature affects the target
3. **Foundation**: It serves as a building block for more complex algorithms
4. **Efficiency**: It requires minimal computational resources
5. **Baseline**: Often used as a benchmark against which to compare more complex models

1.4 Types of Linear Regression

1.4.1 1. Simple Linear Regression

Simple linear regression involves only one independent variable to predict the target variable.

Equation:

$$y = \beta_0 + x_1\beta_1 + \epsilon$$

Use Cases: - Predicting sales based on advertising budget - Forecasting house prices based on square footage - Estimating crop yield based on rainfall

NOTE: Python example in a separate file.

1.4.2 2. Multiple Linear Regression

Multiple linear regression involves two or more independent variables to predict the target variable

Equation:

$$y = \beta_0 + x_1\beta_1 + x_2\beta_2 + ... + x_n\beta_n + \epsilon$$

Use Cases: - Predicting house prices based multiple predictors, such as size, location, age, etc. - Forecasting company revenue based on various economic indicators - Determining product pricing based on multiple factors

NOTE: Python example in a separate file.

1.4.3 3. Polynomial Regression

While not strictly a different type of linear regression, polynomial regression extends linear regression by adding polynomial terms.

Equation:

$$y = \beta_0 + x_1\beta_1 + x_1^2\beta_2 + ... + x_1^n\beta_n + \epsilon$$

Use Cases: - Modeling data with non-linear relationships - Capturing diminishing returns (e.g., advertising spend) - Modeling biological growth patterns

1.4.4 4. Reduced Major Axis (Geometric Mean) Regression

This form of regression is used when the x data has uncertainty. It reduced the distance between the regression line and the data in the orthoganal direction from the line.

$$\beta_1 = sign(r) \cdot \sqrt{\sigma_y^2/\sigma_x^2}$$

To compute the intercept:

$$\beta_0 = \bar{y} - \beta_1 \cdot \bar{x}$$

r correlation between the x and y σ_x Standard deviation of x σ_y Standard deviation of y

1.4.5 5. Regularisation

Regularisation methods use differing penalisation functions, based around 'distance' calculations of the coefficients.

Vector norms Some common distance metrics (vector norms) are list here:

L1 norm (Manhattan distance or Taxicab distance):

$$L_1 = ||\mathbf{v}||_1 = \sum_{i=1}^{n} |v|$$

L2 norm (Euclidean norm):

$$L_2 = ||\mathbf{v}||_2 = \sqrt{\sum_{i=1}^{n} v^2}$$

L infintiy norm:

$$L_\infty = ||\mathbf{v}||_\infty = \max_i |v|$$

1.5 Lasso (*L1*) Regression

Cost Function = RSS + $\times$ (sum of absolute values of coefficients)

$$CostFunction = \sum_{i=1}^{n}(y - \hat{y}_i)^2 + \alpha L_1$$

Where: - α (alpha) is the regularization parameter.

A higher value of α increases the penalty on large coefficients.

Lasso regression needs to solved using a gradient descent type algorithm.

```
from sklearn.linear_model import Lasso
lasso_reg = Lasso(alpha=0.1)
lasso_reg.fit(X, y)
lasso_reg.predict([[1.5]])
```

```
from sklearn.linear_model import SGDRegressor
sgd_reg = SGDRegressor(penalty="l1")
sgd_reg.fit(X, y.ravel())
sgd_reg.predict([[1.5]])
```

1.6 Ridge (*L2*) Regression

Cost Function = RSS + $\alpha \times$ (sum of squared values of coefficients = (L_2^2))
(N.B. the removal of the square root)

$$CostFunction = \sum_{i=1}^{n}(y - \hat{y}_i)^2 + \alpha L_2^2$$

Where: - α (alpha) is the regularization parameter

A higher value of α increases the penalty on large coefficients.

Ridge regression has a closed form. To solve for β:

$$\beta = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{A})^{-1}\mathbf{X}^T\mathbf{y}$$

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

In $\mathbf{A}$ the top left entry is 0 so we do not penalise the offset (intercept) value.

```
from sklearn.linear_model import Ridge
ridge_reg = Ridge(alpha=1, solver="cholesky")
ridge_reg.fit(X, y)
ridge_reg.predict([[1.5]])
```

```
from sklearn.linear_model import SGDRegressor
sgd_reg = SGDRegressor(penalty="l2")
sgd_reg.fit(X, y.ravel())
sgd_reg.predict([[1.5]])
```

1.7 Elasticnet (*L1 & L2*) Regression

$$CostFunction = \sum_{i=1}^{n}(y - \hat{y}_i)^2 + r\alpha L_1 + (1-r)\alpha L_2^2$$

In long form:

$$CostFunction = \sum_{i=1}^{n}(y - \hat{y}_i)^2 + r\alpha \sum_{i=1}^{n}|\beta| + (1-r)\alpha \sum_{i=1}^{n}\beta^2$$

Where: - r is the mix ratio of $L1$ to $L2$

```
from sklearn.linear_model import ElasticNet
elastic_net = ElasticNet(alpha=0.1, l1_ratio=0.5)
elastic_net.fit(X, y)
elastic_net.predict([[1.5]])
```

6. Preprocessing methods Preprocess of the columns (predictor variables) or rows (samples) of the X data can help the modelling process.

Mean centering Subtraction of the mean of a column from each value in the column is called *mean-centering*

$$\mathbf{X}_{\text{centered}} = \mathbf{X} - \mathbf{1}_m \otimes \bar{\mathbf{x}}$$

- $\bar{\mathbf{x}}$ is a vector of the means of each column of $\mathbf{X}$
- $\mathbf{1}_m = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix}$

(N.B. if modelling the intercept, then the column of 1's should remain unchanged)

Variance Scaling (or standardisation) Division of each (mean centred) column by the variance of the column is called *variance scaling*

$$\mathbf{X}_{\text{scaled}} = \frac{\mathbf{X} - \mathbf{1}_m \mu}{\mathbf{1}_m \sigma}$$

Vector normalisation

$$||\mathbf{v}|| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

Division of a row by its length can help scale each sample to the same range.

1.7.1 7. Statistical test metrics

There are many statistical tests for determining a significant result. They depend on specification of a *Null Hypothesis*, which is the statement of no change. e.g. a mean and standard deviation that descibes the data of the population remains the same after a certain treatment. Rejection of the null-hypothesis indicates that the data found after a treatment is statisctically different from the initial population.

1.7.2 Linear regression, Pearson's r coefficient and the R^2 coefficient of determination:

The Pearson's r correlation coefficientit measures the strength and direction of a correlation and is normalised between: - 1 for perfect +ve correlation - 0 for no correlation - -1 for perfect inverse or negative correlation It can be thought of as a normalised correlation.

$$r = \frac{cov(x, y)}{stddev(x) \cdot stdev(y)}$$

$stddev(x) = (\frac{\sum(x-\bar{x})^2}{m-1})^{\frac{1}{2}}$

$cov(x, y) = \frac{\sum(x-\bar{x})(y-\bar{y})}{m-1}$

- x and y are the values of the two variables of interest.
- m is the number of samples of the variables of interest.

R^2 is the Pearsons correlation coefficient squared for a single independent variable, however it is the sum of the variance of each additional independent variable, so the equivalence ends with multiple predictor variables.

R^2 is the measure of variance in the dataset explained by the independent variable.

$$R^2 = 1 - \frac{\sum_i^n (y_i - \hat{y}_i)^2}{\sum_i^n (y_i - \bar{y})^2}$$

- n Is the number of samples (y values)
- $\hat{y}_i$ Is the predicted value of y of sample i
- $\bar{y}$ Is the mean value of y

R^2 is the amount of variance (as a proportion of the total variance) that the independent variable explains in the variation of the depenent variable. Please note,

p-values A p-value is *the probability of rejecting a null-hypothesis that is true.* You want this probability to be small, to say that the data shows a difference due to a treatment.

If a p-value is small, then it is less likely that the Null-hypothesis is true.

One-sample t-Test:

$$t = \frac{\bar{x} - \mu}{s/\sqrt{n}}$$

$\bar{x}$ = sample mean
μ = population mean
s = sample standard deviation
n = sample size

```
from scipy import stats
# Hypothesized population mean
mu_0 = 3.0
# Perform the 1-sample t-test
t_statistic, p_value = stats.ttest_1samp(data, mu_0)
```

Two sample t_Test:

$$t = \frac{\bar{x_1} - \bar{x_2}}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

```
from scipy import stats
# Hypothesized population mean
mu_0 = 3.0
# Perform the 2-sample t-test
t_statistic, p_value = stats.ttest_ind(data1, data2)
```

ref: www.cuemath.com

Confidence intervals on estimates For a choosen significance level, with degrees of freedom $= n - 2$

$$CSS_x = \sum_{i=1}^{n} x^2 - \frac{(\sum_{i=1}^{n} x)^2}{n}$$

$$\Delta\beta_1 = \pm t_{(\alpha/2, n-2)} \sqrt{\frac{s^2}{CSS_x}}$$

$$\Delta\beta_0 = \pm t_{(\alpha/2, n-2)} \sqrt{\frac{s^2 \cdot \sum_{i=1}^{n} x^2}{n \cdot CSS_x}}$$

- CSS_x is the corrected sum of squares
- $\Delta\beta_1$ is the error range of the slope
- $\Delta\beta_0$ is the error range of the intercept
- s^2 is the variance

1.8 Conclusion

Linear regression is a powerful tool for modelling relationships between variables. Its simplicity, interpretability, and efficiency make it an essential algorithm in the machine learning toolkit. While it has limitations in modelling complex, non-linear relationships, variations like polynomial regression can help address some of these limitations.

The choice between simple, multiple, or polynomial regression depends on the specific problem, the relationship between variables, and the complexity of the data.

[ ]: