Projection of a Point on a Line and Applications: Simple Linear Regression

1st Chandrakishor Singh
Department of Information Technology
Indian Institute of Information Technology, Allahabad
mit2021117@iiita.ac.in

2nd Dablu Chauhan
Department of Information Technology
Indian Institute of Information Technology, Allahabad
mit2021115@iiita.ac.in

3rd Madhu Donipati
Department of Information Technology
Indian Institute of Information Technology, Allahabad
mit2021116@iiita.ac.in

***Abstract*—This paper presents a fundamental introduction to projection of a vector onto another vector or line in an n-dimensional space. We begin with a short discussion on projection in general and the circumstances on which this concept helps us in solving a system of linear equations. Then we present some mathematical preliminaries which are required to understand the designing of a projection matrix. This includes the understanding of projection, projection matrix and projection of a point onto a line. We also derive the formula to find the projection in terms of a projection matrix. After that, we represent linear regression as a projection problem where the aim is to minimize the square of the distance between a point and the line which represents the best fit for the given set of points. A theoretical understanding of linear regression along with a demonstration of it is shown with the help of a simple linear regression problem in 2D using python.**

***Keywords*—Projection, Projection Matrix, Linear Regression, Python**

I. Introduction

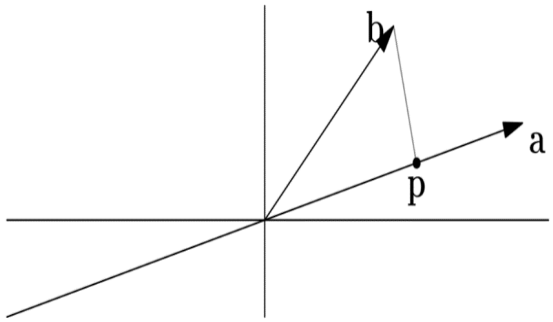
The equation $Ax = b$, as we all know, may not have a solution. Ax is always in A's column space, while b is unlikely to be in A's column space. So, in such situations we may be interested in finding the vector $\hat{x}$ such that it is "closest" to vector b. The "closeness" of the vector $\hat{x}$ to vector b can be defined in various ways. However, typically we want to find the vector $\hat{x}$ such that its euclidean distance from the vector b is minimal. In other words, the sum of squares of the difference between the corresponding components of vectors $\hat{x}$ and b should be as minimal as possible. In such situations, the concept of projection helps in finding such a vector. [1]

II. Mathematical Preliminaries

Before trying to understand projection and its applications like simple regression, one must first understand a few related concepts that are used to obtain the formula for simple linear regression.

A. Projection

Suppose we have a vector b and a line determined by a vector a as follows.



How do we identify the point on the line a that is closest to b ?

This is where the concept of projection comes into play. Above figure shows that the nearest point p is generated by the intersection of a line through b that is orthogonal to a. If we consider p to be an estimate of b, the length of $e = b - p$ represents the approximation's error. We could use trigonometry or calculus to find p, but since we are using linear algebra, why not use it instead! It is also much more convenient. [1]

We know $p = xa$ for some number x since p is on the line through a. We also know that $e = b - xa$ is perpendicular to a:

$$a^T(b - xa) = 0$$

$$xa^Ta = a^Tb$$

$$x = \frac{a^Tb}{a^Ta},$$

and $p = ax = a\frac{a^Tb}{a^Ta}$. Note that, doubling b double p. Doubling a does not affect p.

B. Projection Matrix

As with every other transformation, we would like to express the projection as some matrix transformation. Let

the corresponding transformation matrix be called projection matrix $P : p = Pb$ [2]

$$p = xa = a\frac{a^T b}{a^T a}$$

Hence, the matrix is,

$$P = \frac{a^T b}{a^T a}$$

Note that aa^T is a three by three matrix, not a number; matrix multiplication, as we know, is not commutative. The column space of P is spanned by a because for any b, Pb lies on the line determined by a.

Properties of Projection Matrix:

- Rank of P is 1
- P is symmetric
- $P^2b = Pb$ because the projection of a vector already on the line through a is just that vector.

These properties are true not only for projection matrices in 2 dimensions but also for higher dimensions as well.

A projection matrix contains the information and provides a means of identifying exceptional data points. This approach also simplifies the calculations involved in removing a data point.

III. Linear Regression

A linear technique to model the relationship between a quantity and one or more independent quantities is known as linear regression. When the number of independent variables is one then this linear regression is called simple linear regression. Linear Regression is more versatile and has wide applicability and allows you to understand the strength of relationships between variables. In General, Linear regression is simply the process of picking a line in such a way that the error between the line and the y-coordinate of each data point is minimal. [3]

Here, we will use a simple linear regression model to explain its usage and we will also use it to demonstrate how a line can be fitted onto a given set of data points.

A. Assumptions of Linear Regression

Before understanding the gist of linear regression, let's see what are the assumptions in a linear regression model.

- The Dependent variable is a continuous variable.
- Linear relationship b/w dependent variable and independent variable
  - A linear regression procedure is sensitive to outliers and only works successfully if the predictor or independent variables and the predicted or target variable have a linear relationship. With the use of a scatter plot, this can be clearly detected.
- No Multicollinearity between Independent variables
  - The phenomenon of multicollinearity occurs when two independent variables are substantially associated with one another. To detect the correlation between the independent features the correlation matrix can be utilised. The reason why multicollinearity is not desired is because linear regression is dependent upon the relationships between the predictors and the target variable. It means that a unit change in the independent variable causes the target variable to increase or decrease by the coefficient amount on average. If two independent features have a strong correlation, the algorithm will be unable to create a link between the target and the predictor. [3]
- Normal distribution is followed by Residuals.
- Homoscedasticity
  - The error term, or residuals (the difference between the anticipated and actual value), is constant across all data points and does not follow a specific pattern in their distribution. A scatter plot showing the distribution of residuals vs anticipated values is an excellent indicator of homoscedasticity.
- Residuals should be independently distributed i.e., there should not be any correlation between them

B. Linear Regression as Projection Problem

In simple linear regression, our final goal is to find a line of best fit over the given data points. For that, we try to minimize the square of the distance between a point and the required line. This can be seen as the problem of finding the projection of the data points onto the line such that the error is as minimum as possible.

C. Simple Linear Regression

In a simple linear regression, only one independent variable is present. The aim of the regression is to fit a line $y = \beta_0 + \beta_1 x$ to our data. Here, x is called the independent variable or predictor variable, and y is called the dependent variable or response variable. [4]

Let's take a deeper look at the relevant quantities from the fit before we speak about how to achieve it:

- β_1 is the slope of the line: One of the most significant quantities in any linear regression study is the slope of the line. A score that is very close to 0 suggests that there is little to no link; high positive or negative values, on the other hand, indicate that there is a strong positive or negative relationship.
- β_0 is the intercept of the line

We'll start with a means to quantify how nice a line is before we can actually fit it. Then we'll use that to create the "greatest" line possible.

Proposing a probabilistic model that creates data from lines is one technique to quantify a line's "goodness." The "best" line is then the one for which the data generated by it is "most likely."

IV. A probabilistic model for linearly related data

We know that if two quantities are linearly dependent on each other, then their relationship can be modeled according

to the equation $y = \beta_0 + \beta_1 x$. We'll then add some Gaussian noise. Formally,

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

The noise ϵ_i symbolises the fact that our data will not fully fit the model. We'll use the Gaussian model for the values of ϵ which will vary between 0 and σ^2 in a normal distribution manner. The intercept β_0, β_1 and the noise variance σ^2 is represented as a fixed (i.e., deterministic) but unknown quantity.

A. Solving for the fit: least-squares regression

Assuming that this is actually how the data $(x_1, y_1), ..., (x_n, y_n)$ we observe are generated, then it turns out that we can find the line for which the probability of the data is highest by solving the following optimization problem which will minimize over β_0 and β_1:

$$\sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i)]^2$$

This is known as the least-squares linear regression problem. Given a set of points, the solution is:

$$\hat{\beta_1} = \frac{\sum_{i=1}^{n} x_i y_i - \frac{1}{n}\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x^2 - \frac{1}{n}(\sum_{i=1}^{n} x_i)^2}$$

$$= r\frac{s_y}{s_x}$$

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

B. Linear Regression Demonstration

Here, we will demonstrate how to build a simple linear regression model in python. We'll also plot the result using the popular libraries like matplotlib.

The dataset has been taken from the kaggle. The data is just a toy example and there is no significance as such.

Load data

```
In [2]: train = pd.read_csv('random-linear-regression/train.csv')
        test = pd.read_csv('random-linear-regression/test.csv')

In [3]: train.head()

Out[3]:
```

	x	y
0	24.0	21.549452
1	50.0	47.464463
2	15.0	17.218656
3	38.0	36.586398
4	87.0	87.288984

We have train.csv and test.csv now. We use pandas for reading and manipulating data stored in csv format. We load and read the data (x and y values) in the below snippet. Here, x is the independent variable and y is the dependent variable which will be predicted based on the value of x. Therefore, we have separated x and y values of train and test file into train_x, train_y, test_x and test_y.

Then we train the model. The model is trained using values from the train dataset for x and y. By learning the values of m and c in the straight line or regression line equation, the model learns the optimal fit. Once these values have been learned, the model applies them to the test dataset to make predictions.

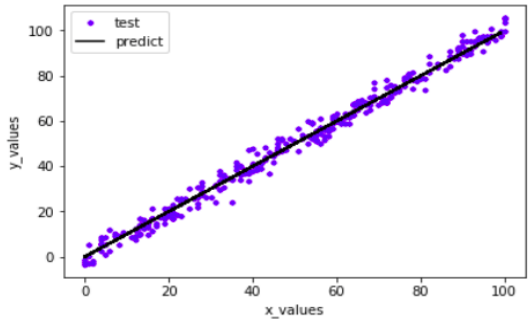
We have defined our own function called linear_reg() which takes the training data as parameters and iteratively minimizes the error using gradient descent. After each iteration we can see that the loss(error) is decreasing.

```
a0, a1 = linear_reg(train_x, train_y)

loss after 0 iterations: 3335.353
loss after 1 iterations: 377.030
loss after 2 iterations: 48.818
loss after 3 iterations: 12.404
loss after 4 iterations: 8.364
loss after 5 iterations: 7.916
loss after 6 iterations: 7.866
loss after 7 iterations: 7.861
```

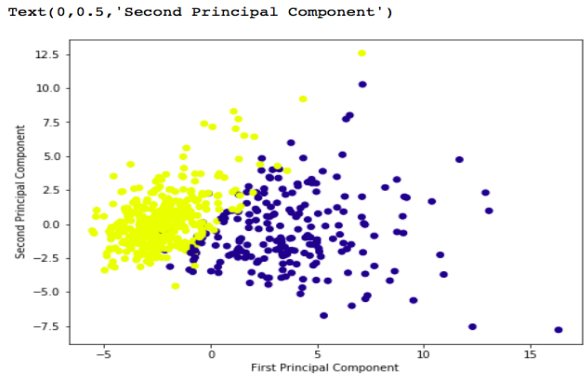
The final simple linear regression model is as follows.

```
visualize_model_predictions(test_x, test_y, a0, a1)
```



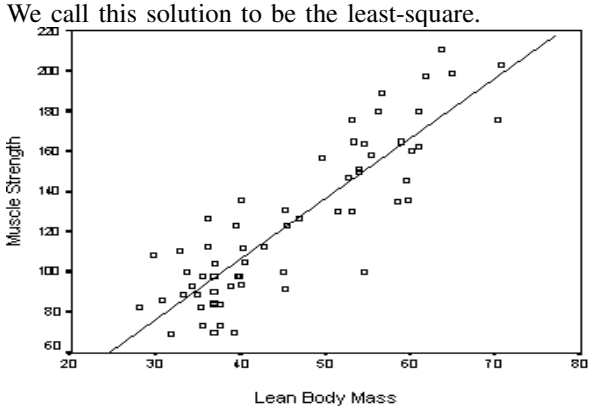
V. Other Applications

- **PCA:** Principal component analysis (PCA) is a fundamental tool in unsupervised learning and data analysis that finds the low-dimensional linear subspace that minimizes the mean squared error between the original data and the data projected onto the subspace. The principal components (PCs) can be obtained by a singular value decomposition (SVD) of the data matrix or eigendecomposition of the data's covariance matrix. PCA is frequently used for dimensionality reduction, feature extraction, and as a pre-processing step for learning and recognition tasks such as classification.
  Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation that converts a set of correlated variables to a set of uncorrelated variables. PCA is the most widely used tool in exploratory data analysis and in machine learning for predictive models



- **Least Square Fitting:** The least-squares method provides the overall rationale for the placement of the line of best fit among the data points being studied. The most common application of this method, which is sometimes referred to as ”linear” or ”ordinary,” aims to create a straight line that minimizes the sum of the squares of the errors that are generated by the results of the associated equations, such as the squared residuals resulting from differences in the observed value, and the value anticipated, based on that model.
  - The least-squares method is a statistical procedure to find the best fit for a set of data points by minimizing the sum of the offsets or residuals of points from the plotted curve.
  - Least squares regression is used to predict the behavior of dependent variables.
  - The least-squares method provides the overall rationale for the placement of the line of best fit among the data points being studied.

  Ex. In real life, there might be times when we want to find a “best-fitting” curve to a set of some given points. For example, people might want to find the relationship between the lean body mass of human bodies and their muscle strength or analyze how the housing price changes by years to obtain an estimation of the price in the future. Most of the time, we might not be able to find such a linear equation that can be satisfied by all of those points, so finding the “closest” solution is the best we can do. We call this solution to be the least-square.



- **CAD/CAM APPLICATION:** Orthogonal projection is an important process in geometric modeling, computer-aided design, and computer graphics. Orthogonal projection of a point in the process of finding a point on a curve or a surface such that the vector connecting the point in space and the point on the curve or the surface becomes perpendicular to the curve or the surface. It is one of the most critical operations in computer-aided geometric design and applications. Orthogonal projection is used as a component for the fitting of geometric features to 2D or 3D point clouds. The fitting error is usually measured using the orthogonal distances from the given points to the geometric features. Minimizing such errors involves satisfaction of the ‘orthogonality’ condition, and the corresponding point on the geometric entity becomes the foot-point for the orthogonal projection. Although the orthogonal projection points may not be computed explicitly. Parameter estimation is the process to estimate parametric values of given points for curve or surface approximation. This is an important step in reverse engineering or object reconstruction when a shape is defined by a set of points. Parametrization is a huge topic in CAD/CAM, Computer Graphics.

VI. Conclusion

This report draws attention towards projection of a point onto a line and projection matrix. We have explained the major prerequisites required to understand projection and how it is used in simple linear regression. We have demonstrated the application of projection by finding a regression line for a random data set in 2D using python programming language.

VII. Acknowledgment

We would like to express our special thanks of gratitude to our teacher Dr. Mohammed Javed, who gave us the assignment which helped me to enhance our understanding in the field of digital image processing and application of linear transformation. We also want to thank Mr. Bulla Rajesh, who mentored us to complete this assignment Successfully.

References

[1] Introduction to Linear Algebra, 5th Edition - MIT Mathematics
[2] Wikimedia Foundation. (2021, November 27). Projection. Wikipedia. Retrieved November 27, 2021, from https://en.wikipedia.org/wiki/Projection.
[3] Banerjee, Sudipto; Roy, Anindya (2014), Linear Algebra and Matrix Analysis for Statistics.
[4] https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/

VIII. Appendix

The python code used is as follows.

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from collections import OrderedDict

# load the data
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

```
# show few data points
train.head()

# get the training data and test data
train_x = train['x']
train_y = train['y']

test_x = test['x']
test_y = test['y']

# print the total number
# of data records of training
# data and testing data
print("Total number of
training examples:%d" %(len(train_x)))
print("Total number of
testing examples:%d" %(len(test_x)))

# Visualize data

# plotting training data
plt.plot(train_x, train_y, 'r.', label = 'train')
plt.xlabel('train_x')
plt.ylabel('train_y')
plt.legend()
plt.show()

# plotting testing data
plt.plot(test_x, test_y, 'b.', label = 'test')
plt.xlabel('test_x')
plt.ylabel('test_y')
plt.legend()
plt.show()

# function to apply the linear regression
def linear_reg(train_x, train_y, lr=0.0001, epochs=10):
    n = len(train_x)

    a0 = np.zeros((n))
    a1 = np.zeros((n))

    itr = 0
    while itr < epochs:
        y_predict = a0 + a1 * train_x
        error = (y_predict - train_y)
        sq_error = error**2
        mean_sq_error = np.sum(sq_error) / n

        d_a0 = 2 * np.sum(error) / n
        d_a1 = 2 * np.sum(error * train_x) / n

        # update the parameters
        a0 = a0 - lr * d_a0
        a1 = a1 - lr * d_a1
                print("loss after %d iterations: %.3f" %(i
                itr += 1

        return a0, a1

# function to visualize the model predictions
def visualize_model_predictions(test_x, test_y, a0
    y_predictions = []
    for i in range(100):
        y_predict = a0 + a1 * i
        y_predictions.append(y_predict)

    plt.plot(test_x, test_y, 'b.', label='test')
    line_values = range(len(y_predictions))
    plt.plot(list(line_values), y_predictions, col

    handles, labels = plt.gca().get_legend_handles
    by_label = OrderedDict(zip(labels, handles))
    plt.ylabel('y_values')
    plt.xlabel('x_values')
    plt.legend(by_label.values(), by_label.keys())
    plt.show()
# get the parameter values by applying the linear
a0, a1 = linear_reg(train_x, train_y)

# plot the regression line
visualize_model_predictions(test_x, test_y, a0, a1
```