📚 Table of Contents


```
begin
    using PlutoUI # visualization purpose
    TableOfContents(title="📚 Table of Contents", indent=true, depth=3, aside=true)
end
```

Lab 01: Linear regression

- Student name: Trần Thuận Phát
- ID: 21127666

How to do your homework

- You will work directly on this notebook; the word **TODO** indicates the parts you need to do.
- You can discuss the ideas as well as refer to the documents, but *the code and work must be yours*.

How to submit your homework

- Before submitting, save this file as `<ID>.jl`. For example, if your ID is 123456, then your file will be `123456.jl`. And export to PDF with name `123456.pdf` then submit zipped source code and pdf into `123456.zip` onto Moodle.

> Note
>
> **Note that you will get 0 point for the wrong submit**.

Content of the assignment:

- Linear regression
- Polinomial linear regression

1. The hypothesis set

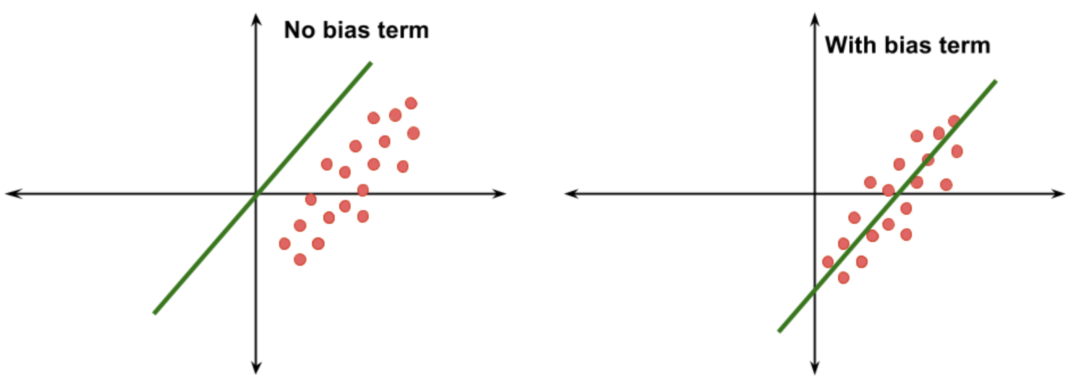
- Linear regression is a linear model, e.g. a model that assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x).
- Generally, a linear model will make predictions by calculating a weighted sum of the input features (independent variables).

$$\hat{y} = w_0 + \sum_{i=1}^{d} w_i x_i$$

  - $\hat{y}$ is the predicted value.
  - d is the number of features.
  - x_i is the i^{th} feature value.
  - w_j is the j^{th} model parameter (including the bias term w_0 and the feature weights $w_1, w_2, \ldots w_d$)
- You can rewrite the first equation in the matrix form as following:

$$\hat{y} = h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \cdot \mathbf{x}$$

  - $\mathbf{w}$ is the model **parameter vector** (including the bias term w_0 and the feature weights $w_1, w_2, \ldots w_n$).
  - $\mathbf{w}^T$ is a transpose of $\mathbf{w}$ (a row vector insteade of column vector).
  - $\mathbf{x}$ is the instance's **feature vector**, *containing* x_0 to x_d, with x_0 *always equal to* 1.
  - $\mathbf{w}^T \cdot \mathbf{x}$ is the dot product of $\mathbf{w}^T$ and $\mathbf{x}$.
  - $h_{\mathbf{w}}$ is the hypothesis function, using the parameters $\mathbf{w}$.



2. Performance measure and the learning goal

- Before we start to train the model, we need to determine how good the model fits the training data. There are a couple of ways to determine the level of quality, but we are going to use the most popular one and that is the **MSE** (Mean Square Error). We need to find the value for $\mathbf{w}$ that will minimize the MSE:

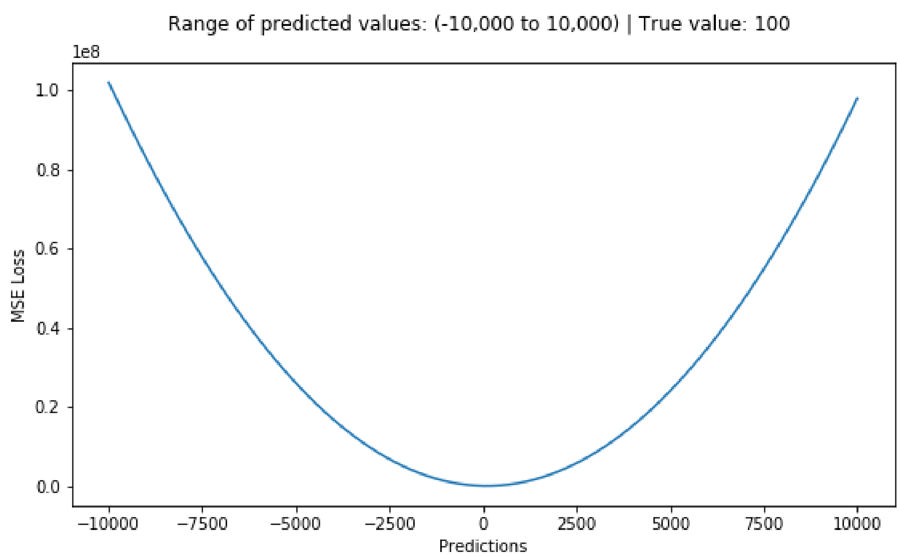
$$\mathbf{w} = \arg\min MSE_{\mathcal{D}_{train}}$$

- MSE on the train set $\mathcal{D}_{train}$ denoted as $(\mathbf{X}, \mathbf{y})$ including n samples $\{(\mathbf{x}_1, y_1), \dots (\mathbf{x}_n, y_n)\}$

$$MSE\left(X, h_{\mathbf{w}}\right) = \frac{1}{m}\sum_{i=1}^{m}\left(\mathbf{w}^T \cdot \mathbf{x}_i - y_i\right)^2$$

$$MSE\left(X, h_{\mathbf{w}}\right) = \frac{1}{m}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

- Example below is a plot of an MSE function where the true target value is 100, and the predicted values range between -10,000 to 10,000. The MSE loss (Y-axis) reaches its minimum value at prediction (X-axis) = 100. The range is 0 to ∞.



- To find the value of $\mathbf{w}$ that minimizes the MSE cost function the most common way (*we have known since high school*) is to solve the derivative (gradient) equation.

$$\hat{\mathbf{w}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

  - $\hat{\mathbf{w}}$ is the value of $\mathbf{w}$ that minimizes the cost function

- In order to reduce the complexity of this approach, I will provide you another version of computing the optimal $\mathbf{w}$:

$$\mathbf{X}\mathbf{w} = \mathbf{y}$$

$$\Leftrightarrow \mathbf{X}^T\mathbf{X}\mathbf{w} = \mathbf{X}^T\mathbf{y}$$

$$\Leftrightarrow \mathbf{w} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$$

3. Implementation

Import library

PlotlyBackend()

```
begin
    using Plots, Distributions
    plotly()
end
```

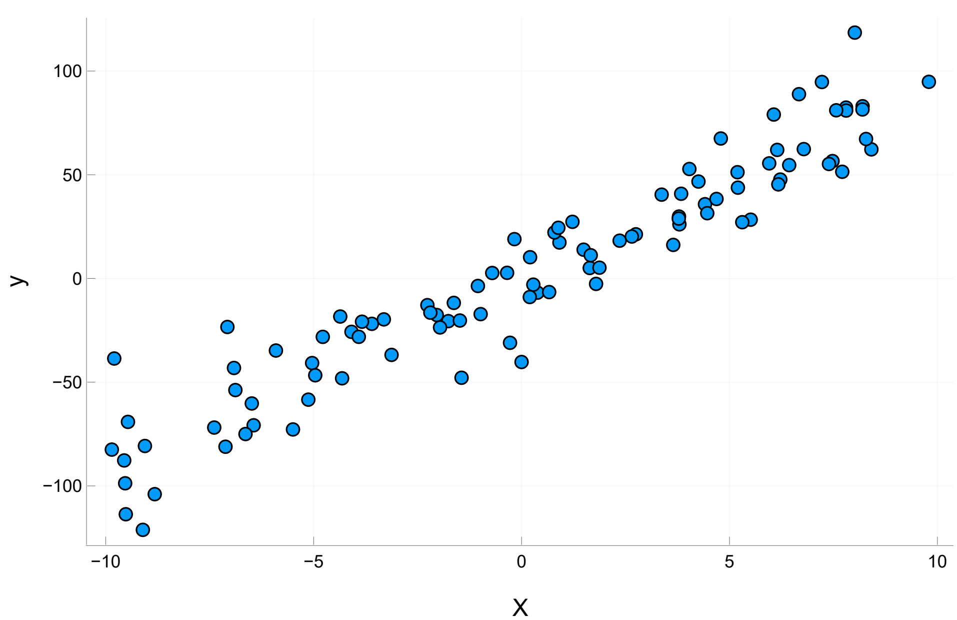
Create data

([2.74879, -0.277598, 5.95524, -6.44322, -4.08998, 3.64738, 8.01314, -2.26494, 5.20209,

```
begin
    function create_data(f, num_data=100)
        # f is the function that is used to generate data
        X = Base.rand(Distributions.Uniform(-10,10), num_data)
        y = f.(X);
        return X, y
    end

    a = Base.rand(Distributions.Uniform(-10,10), 1)[1]
    f(x) = a*x + Base.rand(Distributions.Normal(1,15), 1)[1]
    X, y = create_data(f)
end
```

Visualize data




```
scatter(X, y, label="data points", xlabel="X", ylabel="y", legend=false)
```

TODO:

- Your observations about data:
- Em cảm thấy dữ liệu được phân bổ không giống nhau sau mỗi lần chạy, phân bố một cách rãi rác, ngẫu nhiên và không đều. Tuy nhiên, đa số các dữ liệu này có thiên hướng lên hoặc xuống trãi dài theo một đường thẳng dạng đồ thị tuyến tính bậc 1 với trục X có giá trị nằm trong khoảng từ -10 đến 10.

Training function

```
train_linear_regression (generic function with 1 method)
```

```
function train_linear_regression(X, y)
    """
    Trains Linear Regression on the dataset (X, y).

    Parameters
    ----------
    X : Matrix, shape (n, d + 1)
        The matrix of input vectors (each row corresponds to an input vector);
        the first column of this matrix is all ones (corresponding to x_0).
    y : Matrix, shape (n, 1)
        The vector of outputs.

    Returns
    -------
    w : Matrix, shape (d + 1, 1)
        The vector of parameters of Linear Regression after training.
    """
    # TODO: compute your weight

    w = inv(X' * X) * X' * y

    return w
end
```

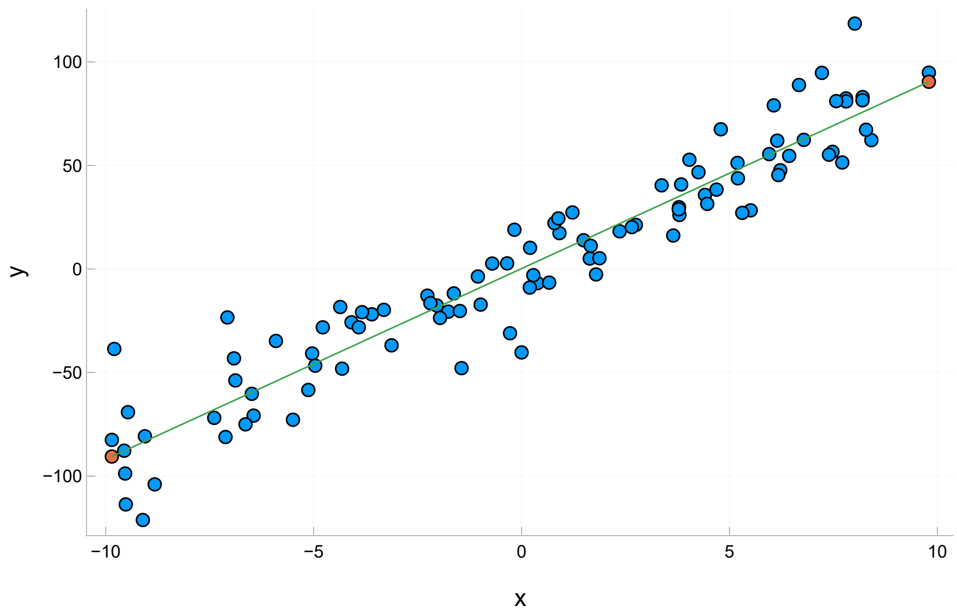
```
begin
    # Construct one_added_X
    # TODO: First column of one_added_X is all ones (corresponding to x_0).

    one_added_X = hcat(ones(Base.size(X, 1) , 1), X)

    println("size of one_added_X = ", Base.size(one_added_X))
    println("size of y = ", Base.size(y))
end
```

```
size of one_added_X = (100, 2)
size of y = (100,)
```

Train our model and visualize




```
begin
    w = train_linear_regression(one_added_X, y)

    # Visualize result
    predicted_ys = one_added_X*w
    scatter(X, y, xlabel="x", ylabel="y", legend=false)
    x_min = minimum(X)
    x_max = maximum(X)
    xs = [x_min x_max]'

    # Construct ones_added_xs
    # TODO: First column of ones_added_xs is all ones (corresponding to x_0).

    ones_added_xs = hcat(ones(Base.size(xs, 1), 1), xs)

    predicted_ys = ones_added_xs*w
    scatter!(xs, predicted_ys, legend=false)
    plot!(xs, predicted_ys, legend=false)

end
```

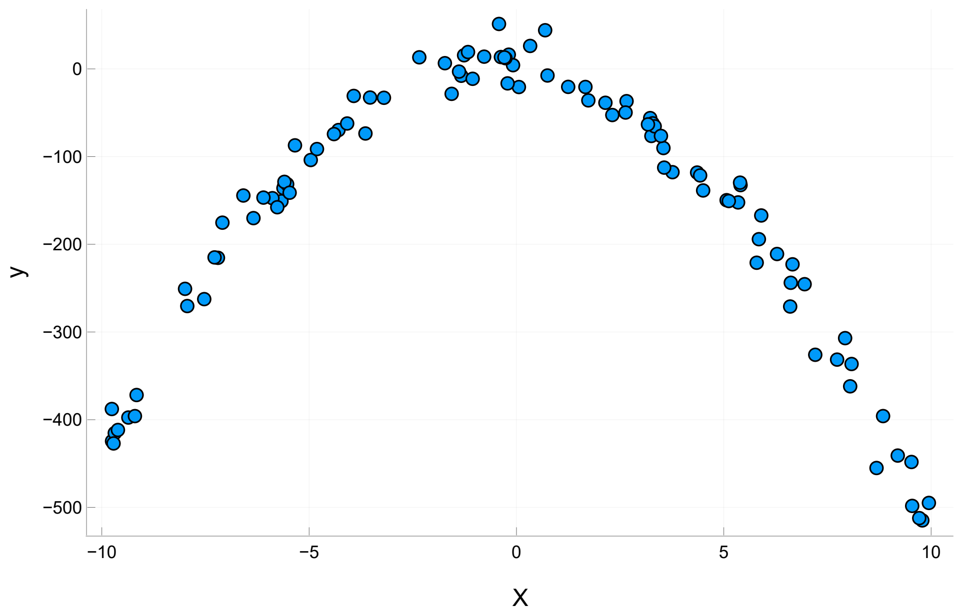
4. Polinomial regression

- Observe the following dataset. Can we use linear regression model to fit this data?

  TODO: Chúng ta không thể sử dụng mô hình hồi quy tuyến tính đối với dữ liệu bên dưới. Bởi vì g(x) là hàm bậc hai của x, có nghĩa là mối quan hệ giữa x và y không tuyến tính dẫn đến mô hình sẽ không nhận biết được độ cong của dữ liệu dẫn đến suy đoán sai lệch. Bên cạnh đó, mô hình hồi quy tuyến tính phù hợp để mô hình hóa mối quan hệ tuyến tính giữa các biến, trong đó mối quan hệ có thể được tính gần đúng như một đường thẳng. Trong tập dữ liệu này, mối quan hệ là bậc hai, do đó mô hình hồi quy tuyến tính sẽ không phù hợp. Thay vào đó, chúng ta sẽ sử dụng mô hình hồi quy đa thức cho dữ liệu bên dưới.

```
([5.34503, -4.96124, 3.54398, 5.06804, 8.84108, -0.185199, -6.343, -0.374948, 9.5435,    m
```

```
begin
    a_ = Base.rand(Distributions.Uniform(-5,5))
    b_ = Base.rand(Distributions.Uniform(-10,10))
    # c_ = Base.rand(Distributions.Uniform(-10,10))
    g(x) = a_*x^2 + b_*x + Base.rand(Distributions.Normal(1,20),1)[1]
    X_, y_ = create_data(g)
end
```



Abstract the problem

- For this kind of datasets, you have to **change the hypothesis set**. For example, in this case, we assume that our data is in the parabol form. Therefore, the hypothesis set will be $\hat{y} = ax^2 + bx + c$. Another assumption: $\hat{y} = ax^3 + bx^2 + cx + d$.
- In general, we have the polinomial regression form:

$$\hat{y} = w_0 + \sum_{i=1}^{d} w_i x_i^i$$

- Re-write the equation above:

$$\hat{y} = h_{\mathbf{w}}(\mathbf{x}) = \mathbf{w}^T \cdot \mathbf{x}, \text{ where } \mathbf{x} = \begin{bmatrix} x_0 = 1 \\ x_1^1 \\ \vdots \\ x_d^d \end{bmatrix} \in \mathbb{R}^{d+1}$$

- To solve this problem, we have to find $\mathbf{w}$ such that:

$$\min_{\mathbf{w}} MSE(X, h_{\mathbf{w}}) = \min_{\mathbf{w}} \frac{1}{m} \|\mathbf{X}\mathbf{w} - \mathbf{y}\|^2$$

- Recall the solution of MSE in section 2:

$$\hat{\mathbf{w}} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \mathbf{y}$$

- Now, it's time for coding!

Solve the problem in code

Step 1: Create polynomial feature

$$\mathbf{X} = \{\mathbf{x}_i = (x_0 = 1, x_1^1, x_2^2, \ldots, x_d^d)\}_{i=1}^{n}$$

```
# TODO: Create X

begin
    function poly_features(X, K)
        # X: inputs of size N x 1
        # K: degree of the polynomial

        X = ones(Base.size(X, 1), 1)

        for i = 1:K
            X = hcat(X.^ i, X)
        end

        return X
    end

    # assume that our data is in form of a parabol
    X = poly_features(X_, 2)
    print(size(X))
end
```

```
(100, 3)
```

Step 2: train our model and find $\mathbf{w}$

```
w =  [-4.87907, -4.43013, 2.26832]
```

```
# TODO: train our model

w = inv(X' * X) * X' * y_
```

Step 3: Visualize our result