Introduction to Machine Learning. Lec.4 Polynomial (?Linear?) Regression

Aidos Sarsembayev, IITU, 2018

Regression is…

- a technique for determining the statistical **relationship between** two or more variables where a change in a dependent variable is associated with, and depends on, a change in one or more independent variables.

http://www.businessdictionary.com/definition/regression.html

Types of regression models

- Simple Linear Regression
- Multiple Linear Regression
- **Polynomial Regression**
- Support Vector Regression (SVR)
- Decision Tree Regression
- Random Forest Regression

Etymology of the term

- **Многочле́н** (или **полино́м** от греч. πολυ- «много» + лат. *nomen* «имя») от n переменных — это сумма одночленов или, строго, — конечная формальная сумма вида

$$\sum_I c_I x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n},$$

Etymology of the term

- The word *polynomial* joins two diverse roots: the Greek *poly*, meaning "many," and the Latin *nomen*, or name. It was derived from the term *binomial* by replacing the Latin root *bi-* with the Greek *poly-*. The word *polynomial* was first used in the 17th century.[1]

- Read more about polynomials: https://www.mathsisfun.com/algebra/polynomials.html

SLR. Formula

Constant

Coefficient

$$y = b_0 + b_1 * x_1$$

Dependent
variable (DV)

Independent
variable (IV)

MLR. Formula

$$y = b_0 + b_1*x_1 + b_2*x_2 + b_3*x_3 + \ldots + b_n*x_n$$

MLR. Formula

Constant

Coefficients

$$y = b_0 + b_1{*}x_1 + b_2{*}x_2 + b_3{*}x_3 + \ldots + b_n{*}x_n$$

Dependent variable (DV)

Independent variables (IVs)

PR. Formula

$$y = b_0 + b_1*x_1 + b_2*x_1^2 + b_3*x_1^3 + \ldots + b_n*x_1^n$$

PR. Formula

Constant

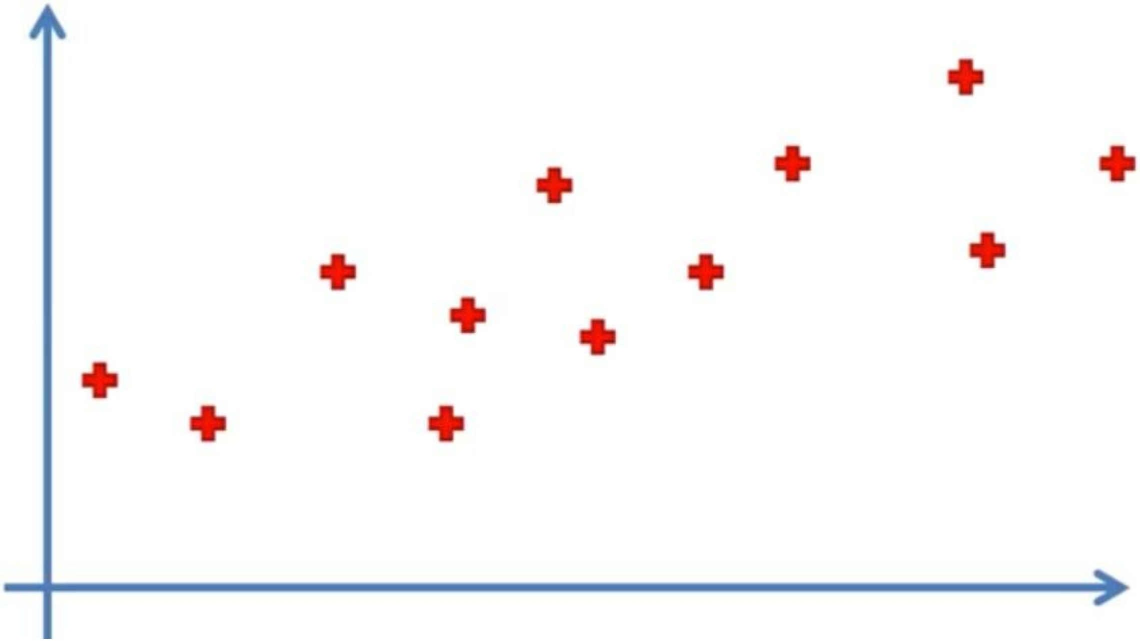
Coefficients

$$y = b_0 + b_1{*}x_1 + b_2{*}x_1^2 + b_3{*}x_1^3 + \ldots + b_n{*}x_1^n$$

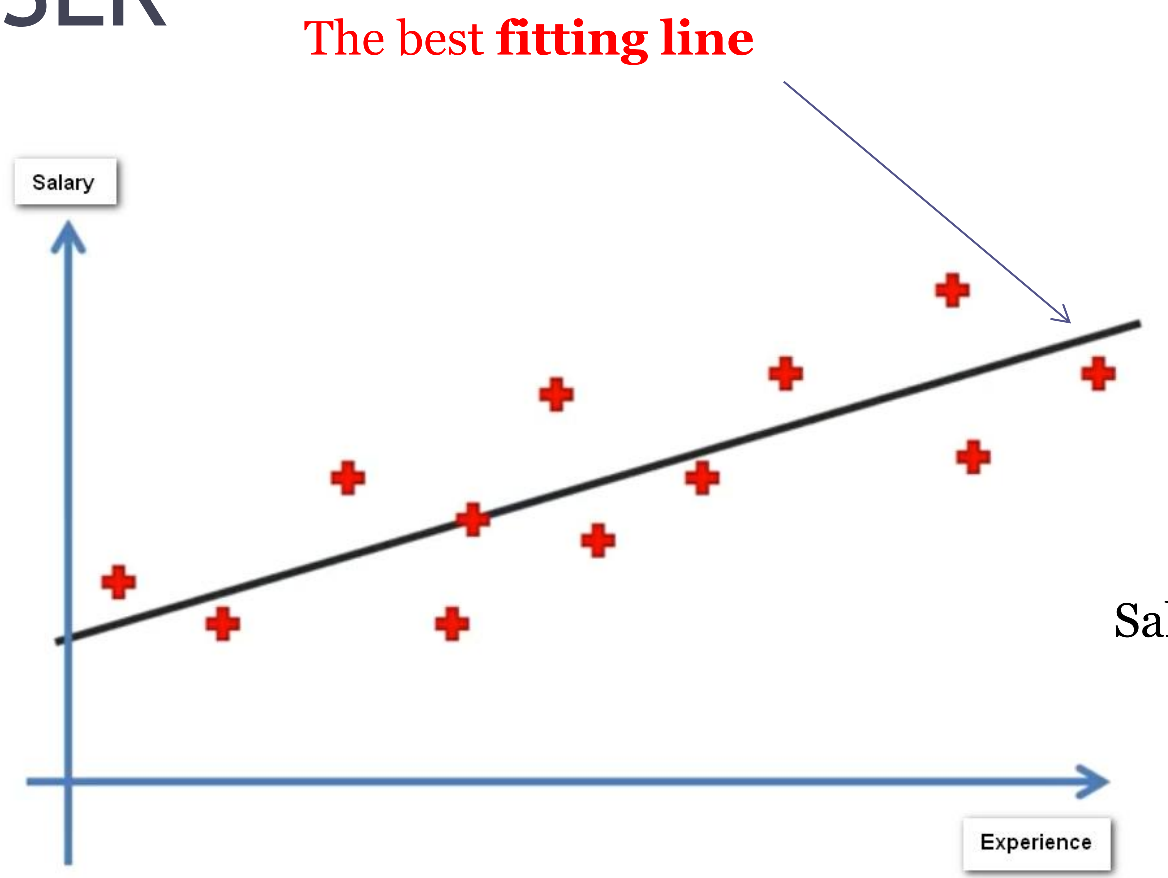
Dependent variable (DV)

Independent variable (IV)

SLR



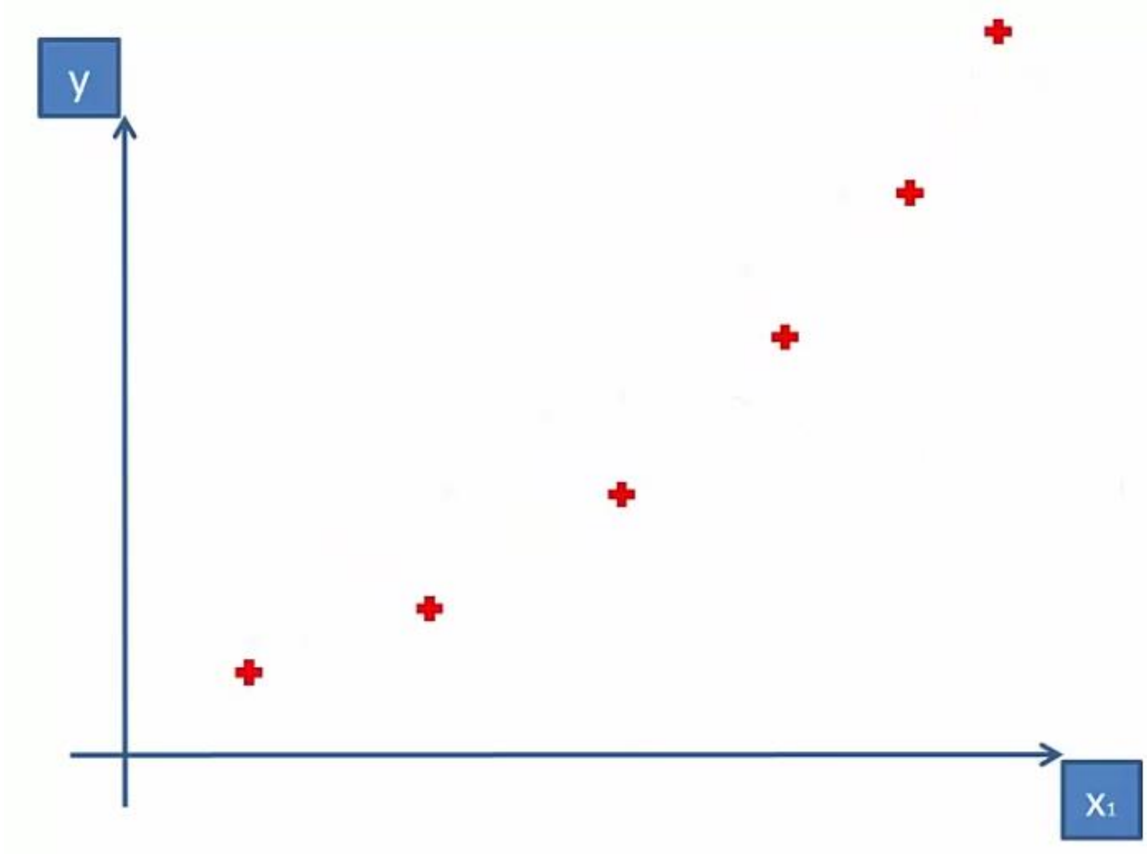
SLR



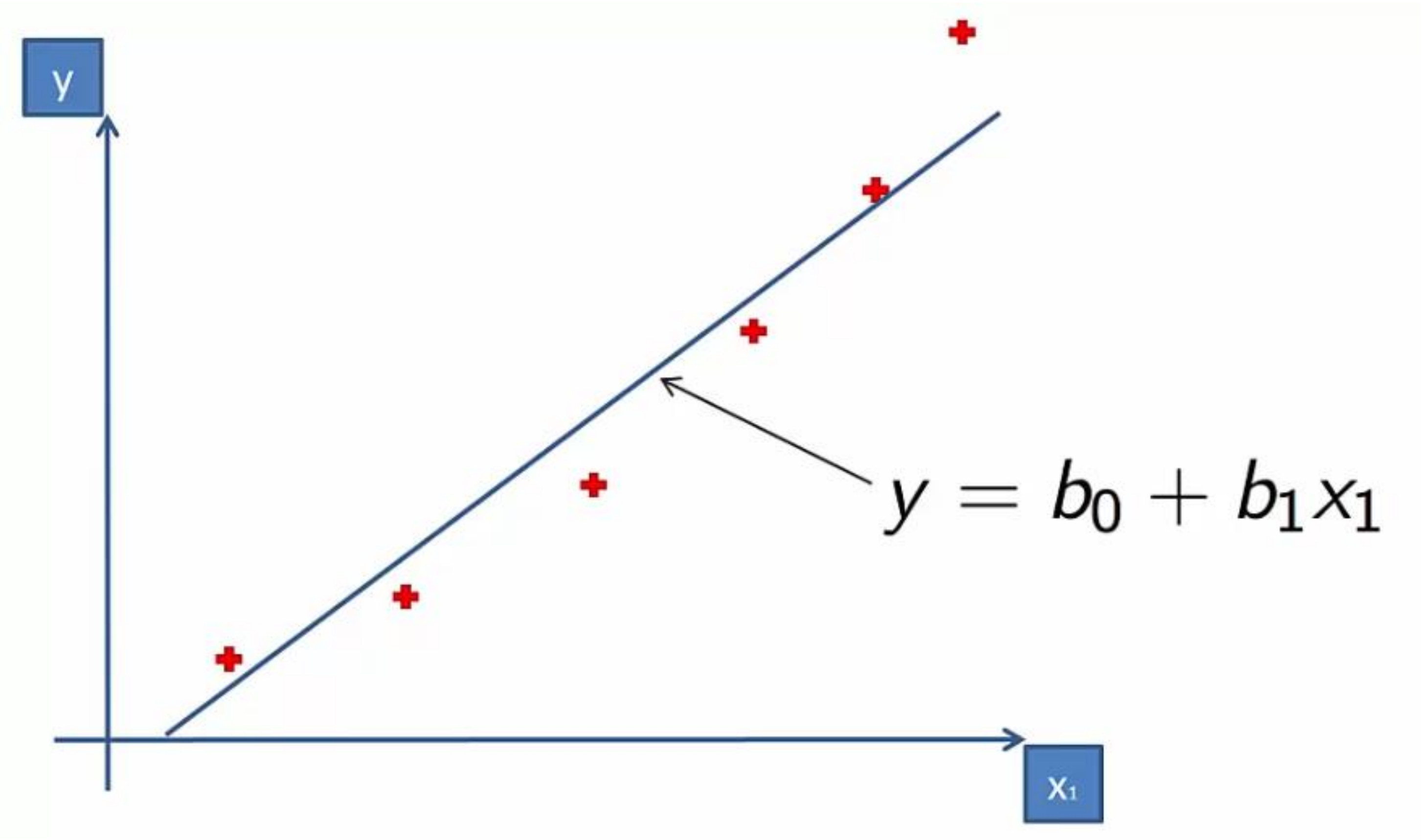
$$y = b_0 + b_1 * x_1$$

$$\text{Salary} = b_0 + b_1 * \text{Experience}$$

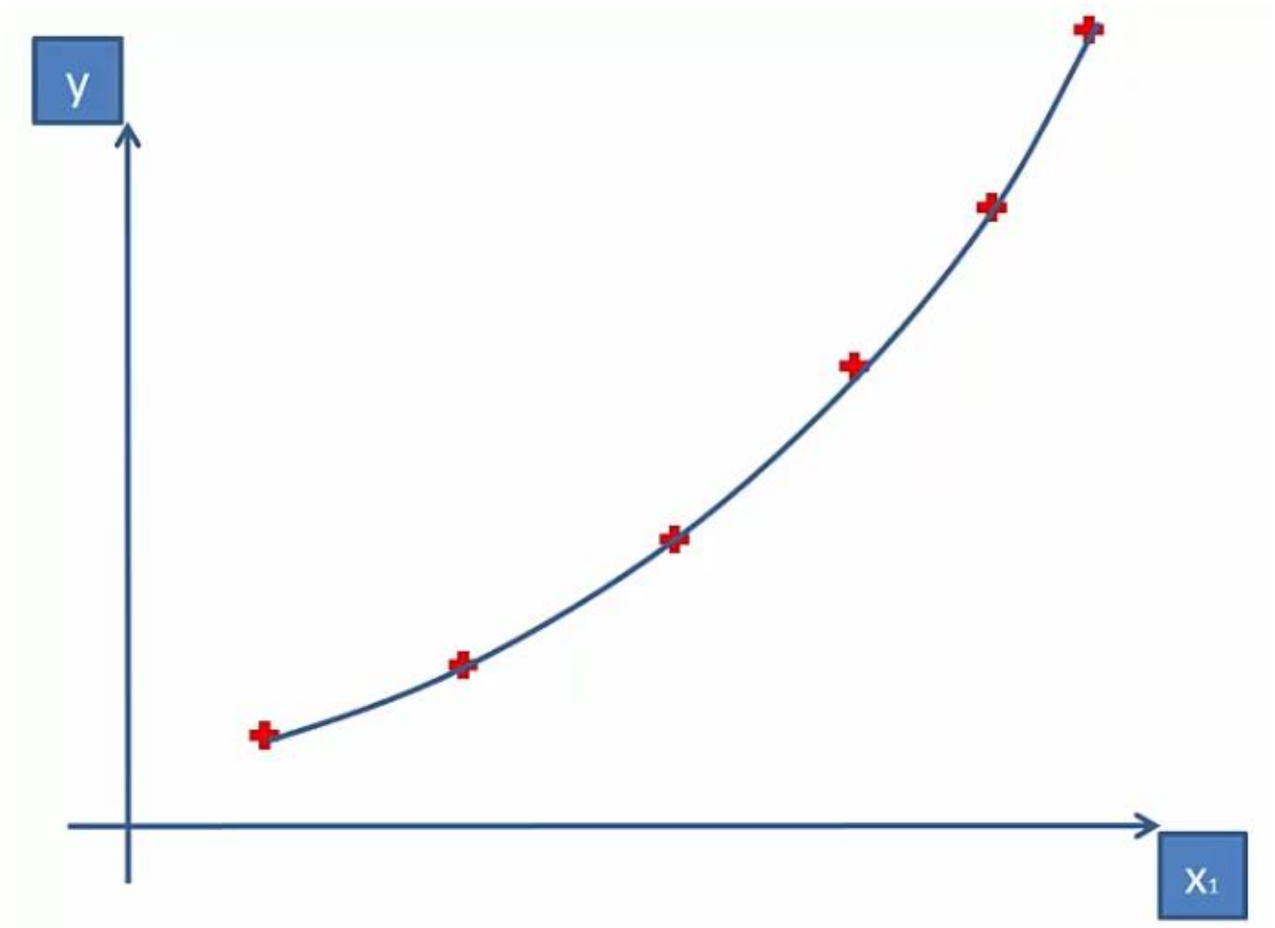
Curve?



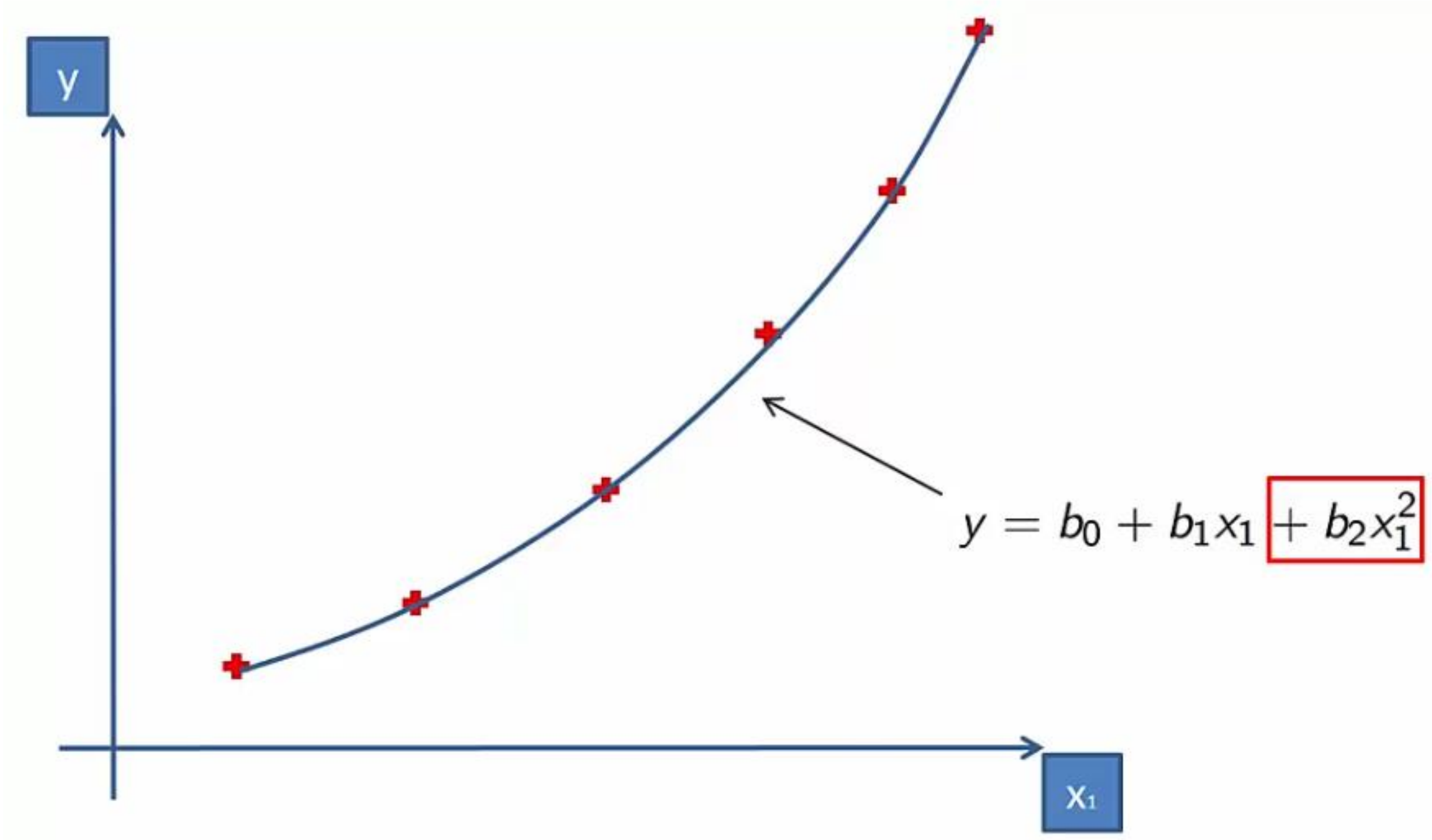
SLR applied



How about this?



PR

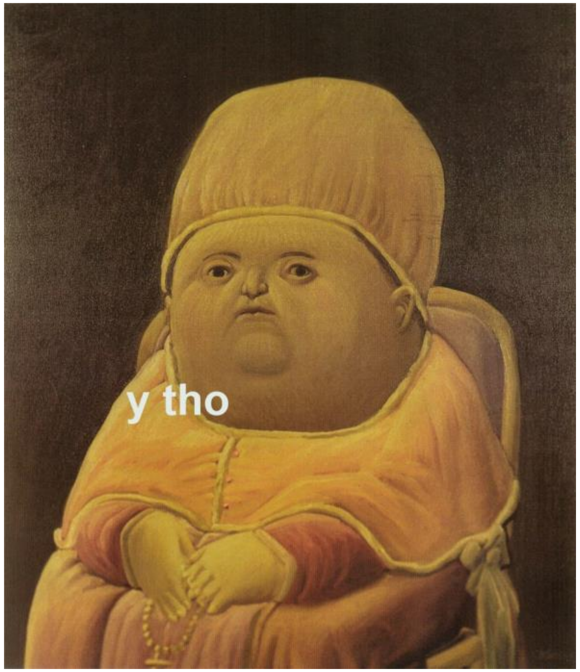


A caveat. Assumptions of LR

- Linearity

Why linear??

- Although this model allows for a nonlinear relationship between Y and X, polynomial regression is still considered as linear regression since it is linear in the regression coefficients, $\beta_1, \beta_2, \ldots, \beta_h$



Throwback to MLR

- The word "linear" in "multiple linear regression" refers to the fact that the model is *linear in the parameters*: $\beta_0, \beta_1, \ldots, \beta_{p-1}$. This simply means that each parameter multiplies an x-variable, while the regression function is a sum of these "parameter times x-variable" terms.
- Each x-variable can be a predictor variable or a transformation of predictor variables (such as the square of a predictor variable or two predictor variables multiplied together).

Throwback to MLR

- The word "linear" in "multiple linear regression" refers to the fact that the model is *linear in the parameters*: $\beta_0, \beta_1, \ldots, \beta_{p-1}$. This simply means that each parameter multiplies an *x*-variable, while the regression function is a sum of these "parameter times *x*-variable" terms.
- Each *x*-variable can be a predictor variable or a transformation of predictor variables (such as the square of a predictor variable or two predictor variables multiplied together).
- Read more about this in: https://onlinecourses.science.psu.edu/stat501/node/311/
- An interesting example of analyzing data before MLR: https://onlinecourses.science.psu.edu/stat501/node/284/

MLR vs PR

- Various predictors {X}
- Various constants
- Doesn't have exponents

- Only one predictor x_1 most of the times (possibly more)
- Various constants
- Has the exponents

PR for two variables

$$y = b_0 + b_1 * x_1 + b_2 * x_2 + b_{11} * x_1^2 + b_{22} * x_2^2 + b_{12} * x_1 x_2$$

Summing it up

- We can call PR as a special case of MLR

- therefore, during the model building process you treat it as MLR (build an instance of LinearRegression class from sklearn)

Model building

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=4)
X_poly = poly_reg.fit_transform(X)
```

Model building

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=4)
X_poly = poly_reg.fit_transform(X)
```

```
poly_reg.fit(X_poly,y)
```

Model building

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=4)
X_poly = poly_reg.fit_transform(X)
```

```
poly_reg.fit(X_poly,y)
```

```
lin_reg2 = LinearRegression()
lin_reg2.fit(X_poly,y)
```

Model building

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=4)
X_poly = poly_reg.fit_transform(X)
```

```
poly_reg.fit(X_poly,y)
```

```
lin_reg2 = LinearRegression()
lin_reg2.fit(X_poly,y)
```

```
lin_ref2_pred = lin_reg2.predict(poly_reg.fit_transform(X))
```

Model building

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=4)
X_poly = poly_reg.fit_transform(X)
```

```
poly_reg.fit(X_poly,y)
```

```
lin_reg2 = LinearRegression()
lin_reg2.fit(X_poly,y)
```

```
lin_ref2_pred = lin_reg2.predict(poly_reg.fit_transform(X))
```

```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_ref2_pred, color = 'blue')
plt.title('Polynomial Regression')
plt.xlabel('Position')
plt.ylabel('Salary')
```

Model building

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=4)
X_poly = poly_reg.fit_transform(X)
```

```
poly_reg.fit(X_poly,y)
```

```
lin_reg2 = LinearRegression()
lin_reg2.fit(X_poly,y)
```

```
lin_ref2_pred = lin_reg2.predict(poly_reg.fit_transform(X))
```

```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_ref2_pred, color = 'blue')
plt.title('Polynomial Regression')
plt.xlabel('Position')
plt.ylabel('Salary')
```

```
lin_reg2.predict(poly_reg.fit_transform(6.5))
```

Model building

```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=4)
X_poly = poly_reg.fit_transform(X)
```

```
poly_reg.fit(X_poly,y)
```

```
lin_reg2 = LinearRegression()
lin_reg2.fit(X_poly,y)
```

```
lin_ref2_pred = lin_reg2.predict(poly_reg.fit_transform(X))
```

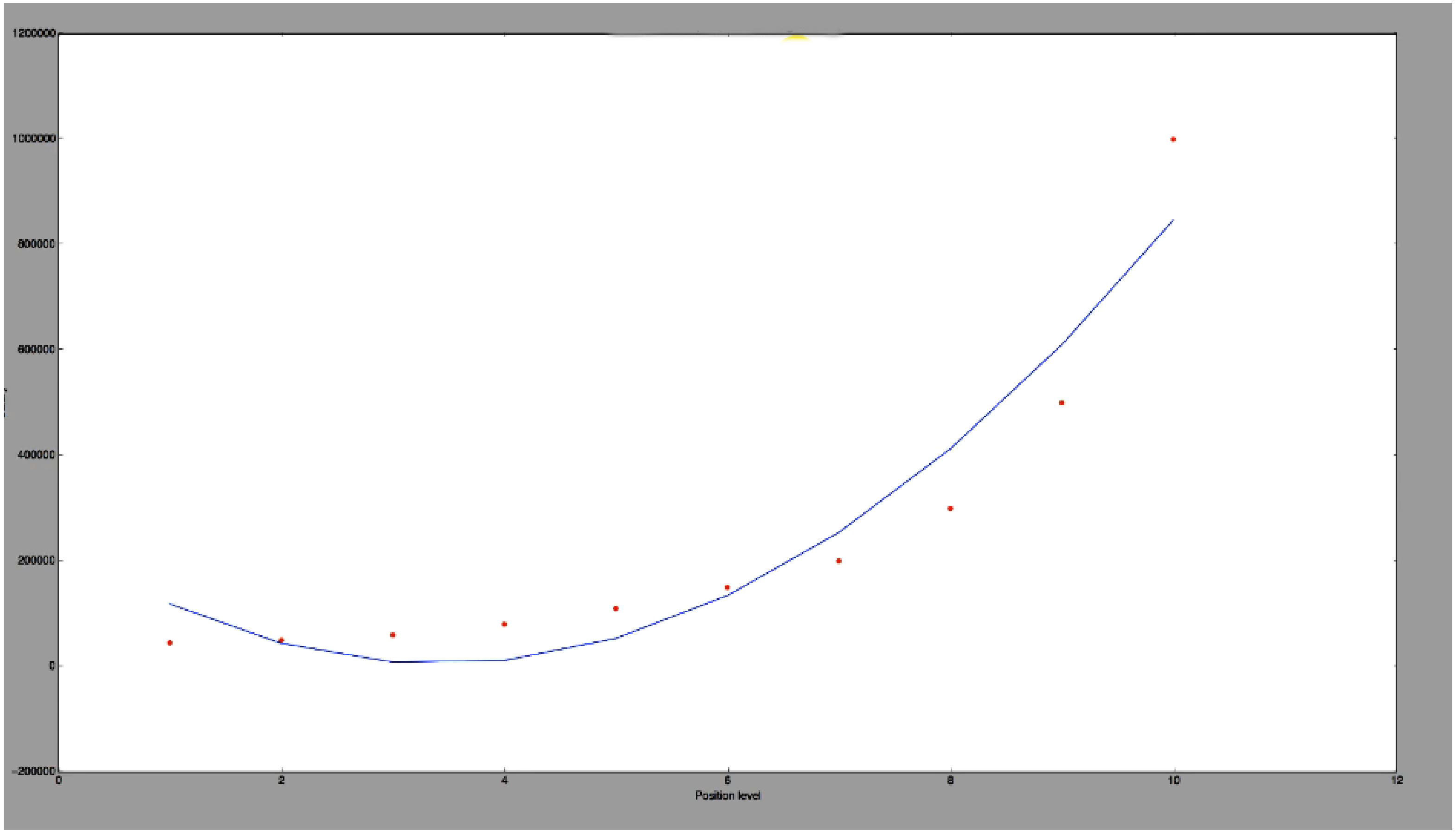
```
plt.scatter(X, y, color = 'red')
plt.plot(X, lin_ref2_pred, color = 'blue')
plt.title('Polynomial Regression')
plt.xlabel('Position')
plt.ylabel('Salary')
```

```
lin_reg2.predict(poly_reg.fit_transform(6.5))
```

```
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape((len(X_grid),1))
#X_grid
```

Instead of the X with the step=1.0
You can have the X with the step = 0.1

The output



1200000
1000000
800000
600000
400000
200000
0
-200000
0
2
4
6
8
10
12
Position level

Datasets sources

- Read more about MLR: https://onlinecourses.science.psu.edu/stat501/node/311/
- An interesting example of analyzing data before MLR: https://onlinecourses.science.psu.edu/stat501/node/284/
- PLR: https://onlinecourses.science.psu.edu/stat501/node/324/