Regression

Richard Xu

August 15, 2022

1 Introduction

This note is to explain the centry-old, simplest regression model: linear and polynomial regression, and some techniques for evaluating regression performance, especially the coefficient of determination (CoD) method.

2 Linear Regression

2.1 explain from what you already studied

remember in the topic of Neural Networks, I described that for any model of scalar output, you need to have:

$$\hat{y_i} = f(\mathbf{x}_i, \theta) \tag{1}$$

and also a loss function $l(\hat{y}_i, y_i)$. So in the case of linear regression:

$$\hat{y_i} = f(\mathbf{x}_i, \theta, b) = \mathbf{x}_i^\top \theta + b \tag{2}$$

It is customary to use $\boldsymbol{\beta}$ instead of θ for linear regression, so we have:

$$\hat{y_i} = f(\mathbf{x}_i, \boldsymbol{\beta}, \beta_0) = \mathbf{x}_i^\top \boldsymbol{\beta} + \beta_0 \tag{3}$$

remember we can do the same trick of extending $\mathbf{x}_i$ by one dimension, and have:

$$\begin{cases} \begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix} & \longrightarrow \tilde{\mathbf{x}}_i \\ \begin{bmatrix} \beta_0 \\ \boldsymbol{\beta} \end{bmatrix} & \longrightarrow \tilde{\boldsymbol{\beta}} \end{cases} \tag{4}$$

therefore, we can rewrite:

$$\begin{aligned} \hat{y_i} &= \beta_0 + \mathbf{x}_i^\top \boldsymbol{\beta} \\ &= \tilde{\mathbf{x}}_i^\top \tilde{\boldsymbol{\beta}} \end{aligned} \tag{5}$$

Since they are just symbols, we can still express them as:

$$\hat{y_i} = f(\mathbf{x}_i, \boldsymbol{\beta}) = \mathbf{x}_i^\top \boldsymbol{\beta} \tag{6}$$

This expression is generic enough and simplify things quite a bit! Since we are trying to minimize the difference between $\hat{y}$ and y in a least square sense, therefore:

$$\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2 \tag{7}$$

The overall loss function would be:

$$\begin{aligned}\mathcal{L} \equiv &= \frac{1}{n}\sum_{i=1}^{n} \ell(\hat{y}_i, y_i) = \frac{1}{2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_i^\top \boldsymbol{\beta} - y_i)^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}(x_{i,1}\beta_1 + \cdots + x_{i,p}\beta_p - y_i)^2\end{aligned} \tag{8}$$

2.2 solving linear regression

from here, we will compute the derivatives with respect to each of β_i, see the $\frac{1}{2}$ comes handy?

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \beta_1} &= x_{i,1}\sum_{i=1}^{n}(x_{i,1}\beta_1 + \cdots + x_{i,p}\beta_p - y_i) = 0 \\ &\vdots \\ \frac{\partial \mathcal{L}}{\partial \beta_p} &= x_{i,p}\sum_{i=1}^{n}(x_{i,1}\beta_1 + \cdots + x_{i,p}\beta_p - y_i) = 0\end{aligned} \tag{9}$$

let's try to solve for simultaneous equations $\hat{\beta}_i = 0$:

$$\begin{aligned}\beta_1\sum_{i=1}^{n}x_{i,1}^2 + \beta_2\sum_{i=1}^{n}x_{i,1}x_{i,2} + \cdots + \beta_p\sum_{i=1}^{n}x_{i,1}x_{i,p} &= \sum_{i=1}^{n}x_{i,1}y_i \\ \beta_1\sum_{i=1}^{n}x_{i,1}x_{i,2} + \beta_2\sum_{i=1}^{n}x_{i,2}^2 + \cdots + \beta_p\sum_{i=1}^{n}x_{i,2}x_{i,p} &= \sum_{i=1}^{n}x_{i,2}y_i \\ &\vdots \\ \beta_1\sum_{i=1}^{n}x_{i,1}x_{i,p} + \beta_2\sum_{i=1}^{n}x_{i,2}x_{i,p} + \cdots + \beta_p\sum_{i=1}^{n}x_{i,p}^2 &= \sum_{i=1}^{n}x_{i,p}y_i\end{aligned} \tag{10}$$

we need some linear algebra to solve this. Notice that each of the line in Eq.(10) gives one row in the linear system of unknowns being $[\beta_1 \dots \beta_p]^\top$:

$$\underbrace{\begin{bmatrix}\sum_{i=1}^{n}x_{i,1}^2 & \sum_{i=1}^{n}x_{i,1}x_{i,2} & \cdots & \sum_{i=1}^{n}x_{i,1}x_{i,p} \\ \sum_{i=1}^{n}x_{i,1}x_{i,2} & \sum_{i=1}^{n}x_{i,2}^2 & \cdots & \sum_{i=1}^{n}x_{i,2}x_{i,p} \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n}x_{i,1}x_{i,p} & \sum_{i=1}^{n}x_{i,2}x_{i,p} & \cdots & \sum_{i=1}^{n}x_{i,p}^2\end{bmatrix}}_{\mathbf{A}\in\mathbb{R}^{p\times p}} \underbrace{\begin{bmatrix}\beta_1 \\ \beta_2 \\ \vdots \\ \beta_p\end{bmatrix}}_{\boldsymbol{\beta}\in\mathbb{R}^{p\times 1}} = \underbrace{\begin{bmatrix}\sum_{i=1}^{n}x_{i,1}y_i \\ \sum_{i=1}^{n}x_{i,2}y_i \\ \vdots \\ \sum_{i=1}^{n}x_{i,p}y_i\end{bmatrix}}_{\mathbf{b}\in\mathbb{R}^{p\times 1}} \tag{11}$$

As long as you have $n \geq p$, the matrix $\mathbf{A}$ is a full rank matrix. You can obtain the solution by solving the linear system:

$$\begin{aligned} \mathbf{A}\hat{\boldsymbol{\beta}} &= \mathbf{b} \\ \hat{\boldsymbol{\beta}} &= \mathbf{A}^{-1}\mathbf{b} \end{aligned} \tag{12}$$

note also that the following is the sample co-variance matrix:

$$\frac{\mathbf{A}}{n} = \frac{1}{n}\mathbf{X}^\top\mathbf{X} = \begin{bmatrix} \frac{1}{n}\sum_{i=1}^n x_{i,1}^2 & \frac{1}{n}\sum_{i=1}^n x_{i,1}x_{i,2} & \dots & \frac{1}{n}\sum_{i=1}^n x_{i,1}x_{i,p} \\ \frac{1}{n}\sum_{i=1}^n x_{i,1}x_{i,2} & \frac{1}{n}\sum_{i=1}^n x_{i,2}^2 & \dots & \frac{1}{n}\sum_{i=1}^n x_{i,2}x_{i,p} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{n}\sum_{i=1}^n x_{i,1}x_{i,p} & \frac{1}{n}\sum_{i=1}^n x_{i,2}x_{i,p} & \dots & \frac{1}{n}\sum_{i=1}^n x_{i,p}^2 \end{bmatrix} \tag{13}$$

2.3 numerical example

let's have the following values:

Table 1: Example of regression data

x_1	x_2	y
2.576	3.625	8.17
5.96	1.399	11.39
1.004	4.822	10.167
3.737	6.764	16.514
3.619	5.563	15.468
5.379	8.236	21.344
7.43	5.283	17.828
4.476	0.374	8.652
0.948	-3.273	-1.785
9.113	6.755	23.287

By the way, I have generated the table values by calling the following Python command (of course, in a real application, the data should be given):

1. generate data where $n = 10$ and $p = 2$, I generate $x_{i,j} \sim \mathcal{N}(\mu = 4, \sigma = 3)$

```
X=np.random.normal(4,3,[10,2])
```

2. next, I append one columns to the left of the data matrix (correspond to β_0):

```
X1=np.append(np.ones([10,1]),X,1)
```

3. next, I generate $\boldsymbol{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$, where I let $\epsilon_i \sim \mathcal{N}(0, 1)$:

```
y=X1@B+np.random.normal(size=10)
```

4. finally, we can append y together to display them in the table

```
np.append(X,y.reshape(-1,1),1)
```

Therefore, we should have data matrix $\mathbf{X}$ to be:

$$\begin{bmatrix} 1.0 & 2.576 & 3.625 \\ 1.0 & 5.96 & 1.399 \\ 1.0 & 1.004 & 4.822 \\ 1.0 & 3.737 & 6.764 \\ 1.0 & 3.619 & 5.563 \\ 1.0 & 5.379 & 8.236 \\ 1.0 & 7.43 & 5.283 \\ 1.0 & 4.476 & 0.374 \\ 1.0 & 0.948 & -3.273 \\ 1.0 & 9.113 & 6.755 \end{bmatrix} \tag{14}$$

I also advice people to set the print option to print limited number of decimal places:

```
np.set_printoptions(precision=3)
np.set_printoptions(suppress=True)
```

We need to compute the following $6 + 3$ quantities:

$$\begin{aligned} \sum_{i=1}^{n} x_{i,1}^2 &= 10 \\ \sum_{i=1}^{n} x_{i,2}^2 &= 258.353 && \texttt{sum(X1[:,1]**2)} \\ \sum_{i=1}^{n} x_{i,3}^2 &= 267.283 && \texttt{sum(X1[:,2]**2)} \\ \sum_{i=1}^{n} x_{i,1}x_{i,2} &= 44.242 && \texttt{sum(X1[:,1])} \\ \sum_{i=1}^{n} x_{i,1}x_{i,3} &= 39.549 && \texttt{sum(X1[:,2])} \\ \sum_{i=1}^{n} x_{i,2}x_{i,3} &= 211.618 && \texttt{sum(X1[:,1]*X1[:,2])} \\ \sum_{i=1}^{n} x_{i,1}y_i &= 131.035 && \texttt{sum(y)} \\ \sum_{i=1}^{n} x_{i,2}y_i &= 713.367 && \texttt{sum(X1[:,1]*y)} \\ \sum_{i=1}^{n} x_{i,3}y_i &= 728.701 && \texttt{sum(X1[:,2]*y)} \end{aligned} \tag{15}$$

looking at Eq.(20), we have the following value for matrix:

$$\begin{bmatrix} \sum_{i=1}^n x_{i,1}^2 & \sum_{i=1}^n x_{i,1}x_{i,2} & \sum_{i=1}^n x_{i,1}x_{i,3} \\ \sum_{i=1}^n x_{i,1}x_{i,2} & \sum_{i=1}^n x_{i,2}^2 & \sum_{i=1}^n x_{i,2}x_{i,3} \\ \sum_{i=1}^n x_{i,1}x_{i,3} & \sum_{i=1}^n x_{i,2}x_{i,3} & \sum_{i=1}^n x_{i,3}^2 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n x_{i,1}y_i \\ \sum_{i=1}^n x_{i,2}y_i \\ \sum_{i=1}^n x_{i,3}y_i \end{bmatrix}$$
$$\implies \begin{bmatrix} 10.0 & 44.242 & 39.549 \\ 44.242 & 258.353 & 211.618 \\ 39.549 & 211.618 & 267.283 \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 131.035 \\ 713.367 \\ 728.701 \end{bmatrix} \tag{16}$$

so finally we should have:

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \begin{bmatrix} 10.0 & 44.242 & 39.549 \\ 44.242 & 258.353 & 211.618 \\ 39.549 & 211.618 & 267.283 \end{bmatrix}^{-1} \begin{bmatrix} 131.035 \\ 713.367 \\ 728.701 \end{bmatrix} = \begin{bmatrix} 1.641 \\ 1.269 \\ 1.479 \end{bmatrix} \tag{17}$$

for the last step, find inverse of a 3×3 matrix can take some time, or you can use Python:

```
A=np.array([[10.0,44.242,39.549],[44.242,258.353,211.618],[39.549,211.
618,267.283]])
b=np.array([131.035,713.367,728.701])
np.linalg.inv(A)@(b)
```

2.4 express in terms of data matrix X

Given a dataset $\mathbf{x}_1, \dots \mathbf{x}_n$, and each $\mathbf{x}_i \in \mathbb{R}^p$. We put each vector of $\mathbf{x}_i$ as a row in the data matrix $\mathbf{X}$:

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{bmatrix} = \begin{bmatrix} — & \mathbf{x}_1 & — \\ \vdots & \vdots & \vdots \\ — & \mathbf{x}_n & — \end{bmatrix} = \begin{bmatrix} x_{1,1} & \cdots & x_{1,p} \\ x_{2,1} & \cdots & x_{2,p} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,p} \end{bmatrix} \tag{18}$$

then we have n corresponding labels, since each label is just a scalar number, then $\boldsymbol{y} \in \mathbb{R}^n$, and $\boldsymbol{\beta} \in \mathbb{R}^p$:

$$\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \qquad \boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} \tag{19}$$

then you probably can tell that matrix from Eq.(20) can be expressed in terms of $\mathbf{X}$:

$$\mathbf{A} = \mathbf{X}^\top \mathbf{X} \qquad \mathbf{b} = \mathbf{X}^\top \boldsymbol{y} \tag{20}$$

You may check with the following Python command:

`X1.T@X1` and `X1.T@y`

to see that the result of data matrix in Eq.(17) indeed agrees with $\mathbf{X}^\top\mathbf{X}$. Therefore, Eq.(20) can be re-written as:

$$\begin{aligned} \mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}} &= \mathbf{X}^\top\boldsymbol{y} \\ \implies \hat{\boldsymbol{\beta}} &= \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\boldsymbol{y} \end{aligned} \tag{21}$$

one can check by calling the following Python command, it agrees with what was previously:

```
np.linalg.inv(X1.T@X1)@(X1.T@y)
```

2.4.1 alternative expression: derivative with respect to vector

In many literature, people tend to use error term $\boldsymbol{\epsilon} \in \mathbb{R}^n$, in which linear model can be expressed in terms of:

$$\begin{aligned} \boldsymbol{y} &= \hat{\boldsymbol{y}} + \boldsymbol{\varepsilon} \\ &= \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon} \\ \implies \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} &= \begin{bmatrix} x_{1,1} & \cdots & x_{1,p} \\ x_{2,1} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,p} \end{bmatrix} \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix} \end{aligned} \tag{22}$$

when you are getting used to performing calculus on matrix and vector directly (which is often the case in machine learning), you can tell that:

$$\begin{aligned} \mathcal{L} &\equiv \frac{1}{2}\sum_{i=1}^{n}(\mathbf{x}_{i,1}\beta_1 + \cdots + \mathbf{x}_{i,p}\beta_p - y_i)^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \\ &= \frac{1}{2}\sum_{i=1}^{n}\epsilon_i^2 \\ &= \frac{1}{2}\|\boldsymbol{\epsilon}\|_2^2 \\ &= \frac{1}{2}\|\mathbf{X}\beta - \boldsymbol{y}\|_2^2 \end{aligned} \tag{23}$$

This means that by minimizing the square of the norm of the vector $\boldsymbol{\epsilon}$, it is equivalent to minimizing in Eq.(8). therefore, let's take derivatives:

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\boldsymbol{\beta}} &= \frac{\partial}{\partial\boldsymbol{\beta}}\frac{1}{2}\|\mathbf{X}\boldsymbol{\beta} - \boldsymbol{y}\|_2^2 \\ &= \frac{\partial}{\partial\boldsymbol{\beta}}\frac{1}{2}(\mathbf{X}\boldsymbol{\beta} - \boldsymbol{y})^\top(\mathbf{X}\boldsymbol{\beta} - \boldsymbol{y}) \\ &= \frac{\partial}{\partial\boldsymbol{\beta}}\frac{1}{2}\Big(\boldsymbol{\beta}^\top\mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} - 2\boldsymbol{\beta}^\top\mathbf{X}^\top\boldsymbol{y} + \boldsymbol{y}^\top\boldsymbol{y}\Big) \\ &= \mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} - \mathbf{X}^\top\boldsymbol{y} \end{aligned} \tag{24}$$

by equating Eq.(25) to zero, we obtained:

$$\begin{aligned} \mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} - \mathbf{X}^\top\boldsymbol{y} &= \mathbf{0} \\ \implies \mathbf{X}^\top\mathbf{X}\boldsymbol{\beta} &= \mathbf{X}^\top\boldsymbol{y} \end{aligned} \tag{25}$$

2.5 Polynomial Regression

Let's look at one-dimensional regression:

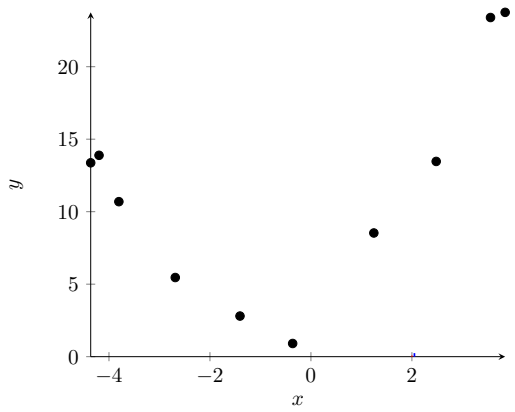

Figure 1: example of x vs y, note they do not form a linear relationship

For each of the data,label pairs (x_i, y_i), we have:

$$y_i \;=\; \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_m x_i^p + \epsilon_i \tag{26}$$

although the model is non-linear, but it is non-linear in terms of x, it is linear in terms of the model parameter $\mathbf{a}$

The model can be written as a system of **linear equations**:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^p \\ 1 & x_2 & x_2^2 & \dots & x_2^p \\ 1 & x_3 & x_3^2 & \dots & x_3^p \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^p \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ a_2 \\ \vdots \\ \beta_p \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix} \tag{27}$$
$$\implies \boldsymbol{y} = \mathbf{X}_{\text{poly}}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

where the data matrix, containing power from $\{0 \dots p\}$ is defined as $\mathbf{X}_{\text{poly}}$:

$$\mathbf{X}_{\text{poly}} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^p \\ 1 & x_2 & x_2^2 & \dots & x_2^p \\ 1 & x_3 & x_3^2 & \dots & x_3^p \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^p \end{bmatrix} \tag{28}$$

Therefore, one can tell that Eq.(27) looks just like Ordinary least squares estimation of linear regression model, in which we can solve $\hat{\boldsymbol{\beta}}$ using Eq.(21):

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}_{\text{poly}}^\top \mathbf{X}_{\text{poly}})^{-1}\, \mathbf{X}_{\text{poly}}^\top \boldsymbol{y} \tag{29}$$

2.6 numerical example

Table 2: Example of polynomial regression data

x	y
-4.197	13.888
-4.362	13.371
-3.804	10.691
-2.687	5.459
-1.405	2.801
-0.362	0.909
1.246	8.531
2.481	13.466
3.557	23.392
3.847	23.748

I have generated $\mathbf{X}_{\text{poly}}$ and $\boldsymbol{y}$ based on an assumed $\boldsymbol{\beta}$ (note again, in real application, you do not know what $\boldsymbol{\beta}$ is):

$$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i \tag{30}$$

I used the following Python command $n = 10, p = 2$ and we are using $\boldsymbol{\beta} = [\beta_0, \beta_1, \beta_2]$:

```
X=np.arange(-5,5,1)+np.random.normal(0,0.5,10)
    X_poly=np.array([X**0,X,X**2]).T
y=X_poly@B+np.random.normal(0,1.0,10)
        np.array([X,y]).T
```

2.6.1 solving numerical example

from this example, we can solve the polynomial regression and find:

$$\begin{aligned}
\hat{\boldsymbol{\beta}} &= (\mathbf{X}_{\text{poly}}^\top \mathbf{X}_{\text{poly}})^{-1}\, \mathbf{X}_{\text{poly}}^\top \boldsymbol{y} \\
&= (\mathbf{X}_{\text{poly}}^\top \mathbf{X}_{\text{poly}})^{-1}\, \mathbf{X}^\top \boldsymbol{y} \\
\text{where } \mathbf{X}_{\text{poly}} &= \begin{bmatrix} 1. & -4.197 & 17.615 \\ 1. & -4.362 & 19.029 \\ 1. & -3.804 & 14.469 \\ 1. & -2.687 & 7.221 \\ 1. & -1.405 & 1.973 \\ 1. & -0.362 & 0.131 \\ 1. & 1.246 & 1.553 \\ 1. & 2.481 & 6.157 \\ 1. & 3.557 & 12.651 \\ 1. & 3.847 & 14.798 \end{bmatrix}
\end{aligned} \tag{31}$$

This is achieved using the following Python command:

```
np.linalg.inv(X_poly.T@X_poly)@(X_poly.T@y)
```

and we obtained the solution:

$$\boldsymbol{\beta} = [3.132 \quad 1.83 \quad 0.997]^\top \tag{32}$$

after we plot it against the data we have:

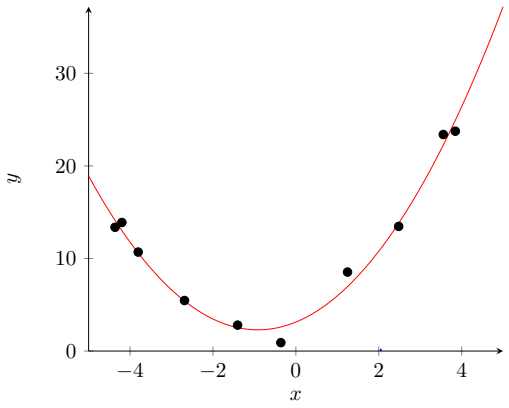

Figure 2: example of x vs y, note they do not form a linear relationship

2.7 Performance of Regression

Remember from the previous lecture, we had developed some effective ways to measure the **overall** performance of the classifiers. However, they are based on "ratios" of correctly matching labels, as they are nominal.

Now we have numeric labels in the regression. We cannot use the same mechanism. Obviously, we can measure the difference between the true label y and the predicted $\hat{y}$ for a single pair of numbers. But how do we measure the overall performance across all labels?

2.7.1 root mean square error (RMSE)

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y-\hat{y})^2} \tag{33}$$

2.7.2 root absolute error (MAE)

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y-\hat{y}| \tag{34}$$

Both RMSE and MAE are self-explanatory. However, they do not use the properties of the true label $\{y_i\}$. Clearly, when the true labels have larger magnitudes (and variances), the difference from the predicted labels should produce larger errors compared with the situation when the true labels have very small magnitudes.

So I'll talk more about the coefficient of determination, which captures the properties/nature of $\{y_i\}$:

2.7.3 Coefficient of Determination (CoD)

Before talking about CoD, I will describe a few terms:

1. let y_i be the ground truth (numerical) labels, then:

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{35}$$

   $\bar{y}$ is just the sample average of all labels

2. total sum of squares: (sample variance of ground truth label $\times$ $N-1$)

$$S_{\text{tot}} = \sum_i (y_i - \bar{y})^2 \tag{36}$$

   this is still using just ground truth labels

3. residual sum of squares (with respect to predicted labels)

$$S_{\text{res}} = \sum_i (y_i - \hat{y}_i)^2 = \sum_i e_i^2 \tag{37}$$

   this is the total sum of square error

4. **coefficient of determination**

$$R^2 = 1 - \frac{S_{\text{res}}}{S_{\text{tot}}} \tag{38}$$

2.7.4 analysis of CoD

1. best case:

$$S_{\text{res}} = 0 \implies R^2 = 1 \tag{39}$$

   this occurs when $e_i = 0\ \forall i$, zero testing error!

2. baseline:

$$S_{\text{res}} = S_{\text{tot}} \implies R^2 = 1 - 1 = 0 \tag{40}$$

   i.e., individual error of S_{res} may vary, i.e., some $(y_i - \hat{y}_i)^2$ may be higher than others. However, their overall sum or square error is bounded and equal to S_{tot}

   if your regression model predicts every data x to $\bar{y}$, i.e., $\hat{y}(x_i) = \bar{y}\ \forall i$, then $R^2 = 0$!

3. worse: negative R^2